VŠB – Technická univerzita Ostrava

Fakulta strojní

Katedra automatizační techniky a řízení

Řízení v reálném čase v prostředí MS Windows pomocí IntervalZero RTX

Real-Time Control in MS Windows by Means of IntervalZero RTX

Student: **Bc. Dušan Kurka**

Vedoucí diplomové práce: **Ing. David Fojtík Ph.D.**

Ostrava 2009

Prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě 23. 05. 2009 ……..

Dušan Kurka

Prohlašuji, že

- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména §35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a §60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB - TUO) má právo nevýdělečně ke své vnitřní potřebě diplomovou (bakalářskou) práci užít (§35 odst. 3).
- souhlasím s tím, že jeden výtisk diplomové práce bude uložen v Ústřední knihovně VŠB - TUO k prezenčnímu nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o diplomové práci budou zveřejněny v informačním systému VŠB - TUO.
- bylo sjednáno, že s VŠB - TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu §12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB - TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě 23. 05. 2009 .....

Dušan Kurka

Vrbka 33,
747 28 Štěpánkovice

ANOTACE DIPLOMOVÉ PRÁCE

KURKA, D. *Řízení v reálném čase v prostředí MS Windows pomocí IntervalZero RTX*. Ostrava: katedra automatizační techniky a řízení, Fakulta strojní VŠB-Technická univerzita Ostrava, 2009, 79 s. Diplomová práce, vedoucí Fojtík, D.

Diplomová práce se zabývá metodikou tvorby aplikací pro doplněk reálného času RTX v operačním systému Microsoft Windows. V úvodu je srovnání operačních systémů reálného času s operačními systémy Microsoft Windows XP a Vista.

Dále je popsán doplněk reálného času RTX, pro systém Microsoft Windows, od společnosti IntervalZero.

V další části je popsán postup a tvorba programů pro doplněk RTX. První z nich je aplikace pro přenos dat na paralelním portu počítače, druhá aplikace ovládá a řídí model levitace feromagnetického předmětu v elektromagnetickém poli.

Zdrojové kódy aplikací jsou přiloženy v příloze diplomové práce.

ANNOTATION OF THESIS

KURKA, D. *Real-Time Control in MS Windows by Means of IntervalZero RTX*. Ostrava: Department of Control Systems and Instrumentation, Faculty of Mechanical Engineering VŠB - Technical University of Ostrava, 2009, 79 p. Thesis, head: Fojtík, D.

Thesis is dealing with methods for creating applications for real time extension RTX in the Microsoft Windows operating system. In the introduction, there is comparison between real time operation systems and the Microsoft Windows operating systems XP and Vista.

In the next part IntervalZero RTX supplement, for Microsoft Windows operating system is described.

The next part is describing a method how to create programs for RTX supplement. The first application is using LPT port of a computer to transfer data ; the second application is controlling a model of levitation of ferromagnetic subject in the electromagnetic field.

The source codes of the applications are attached in the enclosure of the thesis.

Obsah diplomové práce

Seznam použitých symbolů

A/D - analogově číslicový převodník
API - aplikační programové rozhraní (Application Programming Interface)
D/A - číslicově analogový převodník
DCX - modul doplňku RTX pro vzájemnou výměnu dat v reálném čase (Data Control and Exchange)
DLL - dynamicky vázaná knihovna (Dynamic-Link Library)
FIFO - typ datové fronty, položky se zpracovávají podle pořadí jejich vzniku (First In First Out)
HAL - hardwarová abstrakční vrstva oddělující hardware od zbytku operačního systému jednotným softwarovým rozhraním (Hardware Abstraction Layer)
I/O - vstup/výstup (Input/Output)
IPC - jakýkoli způsob přenosu dat mezi procesy běžící v samostatném adresním prostoru (Inter-Process Communication)
IRQ - požadavek na přerušení (Interupt ReQuest)
IST - vlákno obsluhy přerušení (Interrupt Service Thread)
MS Windows - počítačový operační systém vyráběný společností Microsoft
NT - označení technologie architektury operačních systémů MS Windows zcela oproštěné od MS DOSu (New Technology)
OS - operační systém
PC - osobní počítač (Personal Computer)
PID - proporcionálně integračně derivační regulátor (proporcionálně sumačně derivační regulátor), (PSD)
ROM - paměťově čipy, obsah je ukládán převážně jednorázově, to znamená natrvalo (Read Only Memory)
RTOS - operační systém reálného času (Real Time Operating System)
RTSS - subsystém reálného času doplňku RTX (Real Time Sub System)
RTX - rozšiřující doplněk reálného času firmy IntervalZero pro operační systémy NT architektury (Real Time eXtension)
SDK - obsahuje knihovny a utility pro vývojáře v prostředí MS Windows (Software Development Kit)
WDM - rámec pro ovladače zařízení nahrazující starší model VxD (používaný u Win3.11, Win95…) (Windows Driver Model)
a - koeficient filtrace
C - kapacita kondenzátoru [F]
e - regulační odchylka
i - elektrický proud [A]
k - relativní diskrétní čas
k_P - zesílení regulátoru
L - indukčnost vinutí [H]
L_∞ - Konstanta dána fyzikálními vlastnostmi vinutí, jádra a levitujícího předmětu [H]
m - hmotnost levitujícího předmětu [kg]

Q	-	konstanta dána fyzikálními charakteristikami vinutí, jádra a levitujícího předmětu [H·m]
R	-	elektrický odpor vinutí [Ω]
T	-	vzorkovací perioda [s]
T_D	-	derivační časová konstanta [s]
T_f	-	časová konstanta filtru [s]
T_I	-	integrační časová konstanta [s]
u	-	akční veličina, popřípadě elektrické napětí [V]
w	-	žádaná veličina
X_∞	-	konstanta dána fyzikálními charakteristikami vinutí, jádra a levitujícího předmětu [m]
x	-	vzdálenost mezi levitujícím objektem a magnetem [m]
y	-	regulovaná veličina

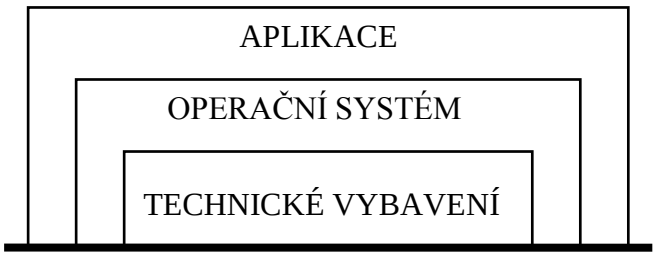
1. Úvod

Operační systémy nás v dnešní době obklopují na každém kroku. V jistých úpravách se operační systémy postupně stávají součástí téměř každého elektronického zařízení, které máme v domácnosti, které používáme pro uspokojení našich potřeb, ať už si připravujeme jídlo v mikrovlnné troubě, telefonujeme se svými kamarády mobilním přístrojem, sledujeme televizní pořady přes multimediální centrum, které máme umístěno v domácnosti. Ale nejedná se jen o systémy, které nalézáme v našich domácnostech.
Díky tomu, že se stávají tyto systémy tak rozšířené, je zcela zákonité, že mnoho uživatelů vyžaduje, aby je mohly používat v co možno nejjednodušší formě a v co možno nejširším záběru, aby mohly ze svého obývacího pokoje v průběhu sledování televize komunikovat se svými přáteli přes komunikační programy, aby mohli při nepřítomnosti sledovat majitelé, přes bezpečnostní kamery a senzory, zda jejich domácnost či podnik nenavštívil lapka a zda je vše v podniku nebo domácnosti v pořádku. Aby si uživatelé mohli nastavit teplotu v domácnosti, když se vracejí domů z práce atd.
Takovéto systém, které řídí a přes které se přistupuje do domácnosti nebo společnosti, musí vyhovovat v mnoha směrech a požadavcích. Systém musí dokázat spravovat všechna tato zařízení, která jsou k němu připojena, a při požadavcích na tento systém je nutné, aby byl systém schopen práce s vysokou přesností, a musí být maximálně spolehlivý. Současné běžně užívané operační systémy tyto požadavky splňují jen z části a specializované operační systémy jsou pro běžné uživatele příliš drahé. Pro vytvoření opravdu spolehlivého systému je nutné tyto systémy obohatit. K takovým to účelům je vytvořen společností IntervalZero doplněk RTX, který se instaluje do operačního systému Microsoft Windows.
Diplomová práce se zabývá doplňkem RTX, který rozšiřuje operační systém Microsoft Windows o reálný čas, jehož autorem je společnost IntervalZero. Tento doplněk je instalován do operačního systému Windows (NT, 2000, XP, Vista). Čímž se z těchto operačních systémů stávají platné systémy reálného času. Takto upravený systém nám následně umožňuje řídit procesy v reálném čase.

2. Operační systém

Operační systém patří mezi tzv. systémový software a hlavním úkolem operačního systému je zabezpečit běh a programovou podporu aplikačních programů [WIKIPEDIE 2008].

Jeho hlavní náplní je překrývat různé verze a implementace hardware (od různých výrobců) tak, aby pro program, který využívá služeb operačního systému, používal jednotný přístup ke službám operačního systému nezávisle na použitém hardware. Dále operační systém poskytuje různé služby, které podporují snazší implementaci aplikačních programů, např. služby souborového systému, síťové služby, apod.



Obrázek 1. Postavení operačního systému v počítači [WIKIPEDIE 2008]

Cílem je, aby aplikace neměly přímý přístup k technickému vybavení PC, čímž se zvyšuje stabilita a bezpečnost.

Obrázek 2. Obecná struktura operačního systému [WIKIPEDIE 2008]

Obecná struktura operačního systému je tvořena:
Jádrem což je základní modul operačního systému – jenž po zavedení do paměti řídí činnost počítače, poskytuje procesům služby, řeší správu prostředků a správu procesů.

Ovladači, což jsou zvláštní “moduly” obvykle dodávané výrobci konkrétních zařízení, jenž implementují tato zařízení do OS. Použití systému s ovladači umožňuje snadnou konfigurovatelnost a různou rozsáhlost technického vybavení pod stejným operačním systémem.

Příkazový interpretem, programem, který umožňuje uživatelům zadávat příkazy ve speciálním, obvykle jednoduchém jazyce.

Podpůrnými programy sestavující programy a překladače (C v OS UNIX), které jsou na stejné úrovni jako aplikační programy.

2.1. Základní úkoly operačního systému

Operační systém:

- organizuje přístup a využívání zdrojů počítače,
- fyzicky zajišťuje vstup a výstup dat podle požadavků ostatních programů,
- komunikuje s uživatelem a na základě jeho pokynů vykonává požadované akce,
- reaguje na chybové stavy programů a špatné požadavky uživatelů tak, aby tyto chyby nezpůsobily zásadní destrukci systému nebo poškození dat,
- eviduje využívání systémových zdrojů,
- zajistit vhodné prostředí pro spouštění programů,
- provádět správu hardwarových zdrojů, jako je paměť, centrální procesorová jednotka a periferní zařízení.

Obecný účel operačního systému je snížit náklady na provoz počítače na minimum:

1. maximálně ulehčit práci aplikačním programátorům,
2. zvýšit využití všech částí počítače na maximum,
3. zajistit maximální bezpečnost ukládaných a zpracovaných dat.

Operační systémy můžeme rozdělit podle požadavků na odezvu následovně:

1. dávkový - úlohy se zadávají jako po sobě jdoucí příkazy a uživatel musí čekat, dokud není splněna celá úloha,
2. interaktivní - dovolují uživatelům reagovat na úlohy (Windows, Linux...),
3. real – time - specializované operační systémy, často organická součást zařízení, stroje, technologie, při real-time řízení je větší náročnost na paralelní běh výpočetních procesů, ovládání nestandardních periferií v řízené soustavě, nejsou potřebné funkce vytvářející uživatelské rozhraní.

3. Systém reálného času a operační systém reálného času

Řada systémů vyžaduje zpracování svých požadavků v tzv. kritickém čase, jako např. v telekomunikacích, lékařství, robotice, letectví, vojenství atd. Všeobecně se takovýmto systémům říká systémy reálného času.
Musíme rozlišovat mezi systémem reálného času a operačním systémem reálného času (RTOS). Reálný čas systému závisí na komponentách celého systému. Zatím co RTOS je jen jedna část v systému reálného času [FOJTÍK 2004].

3.1. Systémy reálného času

Jsou systémy, které vyžadují okamžité zpracování požadavků v kritickém čase. Využívané jsou především v lékařství, letectví, automobilovém průmyslu, v robotice. Tyto systémy musí plnit vysoké nároky na rychlost a čas, ve kterém se musí požadované úkoly splnit.
Jedna z nejpoužívanějších definic systému reálného času:
"Systém reálného času je systém, ve kterém je správné počítání závislé nejen na bezchybném logickém výpočtu, ale také na čase, ve kterém je vyřešen [COMP.REALTIME 2004].“

3.2. Operační systémy reálného času (RTOS)

Nejčastější definice operačního systému reálného času je:
"RTOS je systém, který zpracovává libovolnou možnou událost nebo kombinaci událostí předem determinovatelným způsobem." Determinovatelným způsobem znamená, že je dána posloupnost zpracování jednotlivých událostí pomocí priorit a tak je možné pro kteroukoliv událost určit maximální čas nutný k jejímu zpracování [COMP.REALTIME 2004].

Požadavky, které operační systém reálného času musí splňovat:

- preemptivní multitasking,
- přidělování času procesoru početním vláknům podle priorit,
- řešení problému "inverze priorit",
- predikovatelné možnosti synchronizace běžících vláken.

Operační systém reálného času musí být předvídatelný:

- musí být známá maximální doba, během které jsou maskována přerušení operačním systémem a ovladači zařízení,
- prodleva přerušení (doba od přerušení k následnému běhu vlákna) musí být předvídatelná a kompatibilní s požadavky aplikace,
- doba každého systémového volání by měla být předvídatelná a nezávislá na počtu objektů v systému.

Následující rozdělení operačních systémů reálného času je na soft a hard RTOS. Toto rozdělení je podle času uplynulého od vzniku do zpracování požadavku.

3.3. Hard RTOS

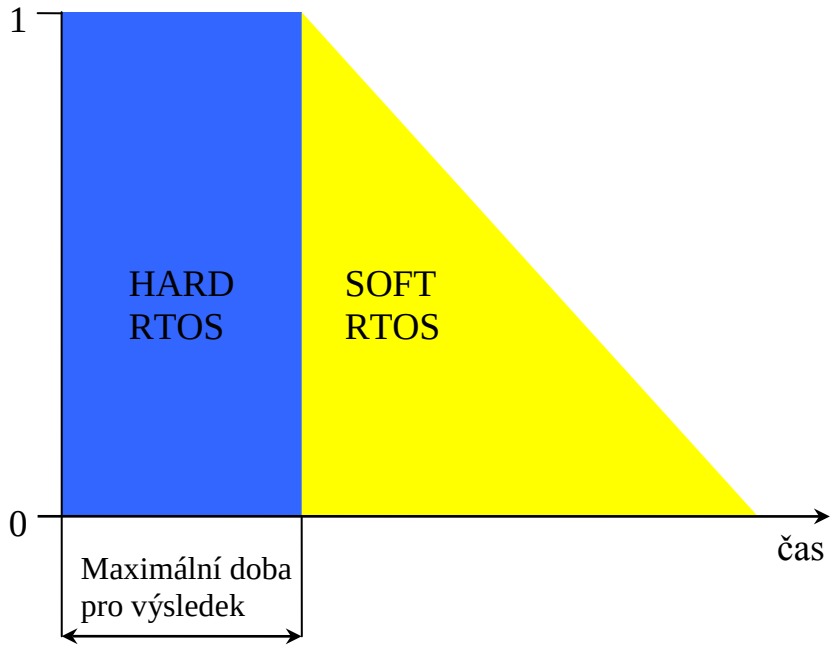
Hard RTOS se používá v systémech, ve kterých se nedodání výsledku v určitém časovém intervalu projeví úplným selhání systému. Pokud tedy výsledek není dodán v daném limitu, je výsledek stejný, jako kdyby nebyl dodán vůbec. Mezi RTOS patří OpenControl (Nematron Software), Paradym – 31(Intellution), Windows CE, PharLap ETS atd.

Tvrdé přidělování času vyžaduje, aby plánovač procesu věděl přesně, jak dlouho trvá která operace, každé operaci je potom garantována maximální přípustná doba. To je ovšem nemožné u systému např. s virtuální pamětí, protože tyto systémy vyvolávají nevyhnutelně a nepředvídatelně změny v množství času nutného k průběhu procesu. Proto je toto přidělování času užíváno u speciálního softwaru spuštěného na vyhrazeném hardwaru, kde není třeba všech schopností moderního OS (operační systém) [FOJTÍK 2004].

3.4. Soft RTOS

Soft RTOS se používá v systémech, ve kterých se nedodání výsledku v určitém časovém intervalu projeví jen zhoršením kvality daného procesu a nezpůsobí havárii systému. Dodržení požadované lhůty dodání výsledku je u soft RTOS okolo 95 %. Mezi soft RTOS patří BridgeVIEW (National Instruments), SoftLogix 5, WinAC (Siemens) atd.

Měkké přidělování času (soft real-time computing) je méně omezující. Vyžaduje pouze, aby kritické procesy získaly vyšší prioritu než ostatní. Z toho vyplývá, že přidání měkkého přidělování času do systému sdílení času může vyvolat nespravedlivé alokace zdrojů, může vést k delším čekacím dobám nebo k možnosti umoření procesu. Výsledkem je však univerzální systém podporující multimedia, vysokorychlostní interaktivní grafiku a mnoho dalších úloh, které by nemusely být akceptovatelné v jiném systému bez podpory měkkého přidělování času [FOJTIK 2004].



Obrázek 3. Důležitost dodání výsledku v čase u hard a soft RTOS [FOJTÍK 2004]

4. RTOS vs. MS Windows XP/Vista

Jedním ze základních předpokladů operačních systémů reálného času je přítomnost rychlého a přesného časovače. V operačním systému MS Windows XP/Vista máme několik nástrojů, například Multimediální časovač [Microsoft 2003], ten je jeden z nejpřesnějších a částečně splňuje kritéria.
S tímto časovačem lze zajistit periodickou činnost s frekvencí až 1 kHz. Ale tato frekvence je dosažitelná jen za podmínky relativně nezatíženého systému.
V případě zatížení systému se může periodická aktivita zpozdit až o stovky milisekund [FOJTÍK 2004].
Z toho vyplývá, že časovače obsažené v MS Windows XP/Vista nejsou vhodné pro systémy reálného času.

4.1. Windows XP/Vista a systém reálného času

Windows XP/Vista nejsou v základním provedení operačními systémy reálného času. Byly vytvořeny jako operační systémy pro obecné účely, vhodné jak pro interaktivní systém na počítačích typu desktop, tak i jako serverové systémy v síti. Nedostatky Windows XP/Vista spočívají zejména v následujících příčinách:

- příliš málo priorit vláken,
- nedeterministická rozhodnutí plánovače,
- řešení vzniku inverze priorit, zvláště při zpracování přerušení.

4.2. Výběr nejznámějších OS reálného času

Operační systémy reálného času jsou nejčastěji využívány ve specifických zařízeních, a takzvaně „šita zařízení na míru“.

4.2.1. QNX

QNZ je hard real-time operační systém, který vyvíjí od roku 1980 firma QNX Software Systems v Kanadě. Jedná se o real-time operační systém, postavený na architektuře mikrojádra. Operační systém, včetně shellu a externích programů je unixového typu. Mezi podporované platformy patří x86, PowerPC, MIPS, ARM, StrongARM, XScale.

Obrázek 4. Logo QNX RTOS

Mikrojádro využívá jen několik nezbytných serverů pro správu paměti, přidělování procesů. Každý proces má přidělenu vlastní část paměti, takže zhroucení systému je velmi nepravděpodobné. Mezi významná pozitiva tohoto systému patří, že jádro je možno restartovat za běhu systému.
Do systému jsou integrovány protokoly TCP/IP, PPP, DHCP, NFS, RCP.
QNX je vybaven grafickým rozhraním Photon microGUI, které má jen minimální nároky na paměť a je plně konfigurovatelné.

4.2.2. RT - Linux

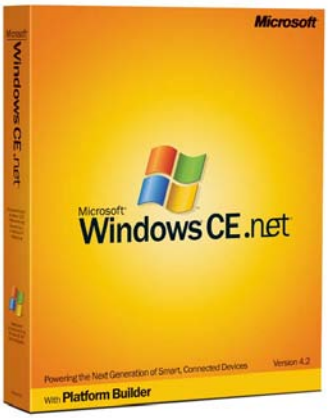
RTLinux je hard real-time varianta Linuxu, která umožňuje ovládání robotů, data pořizujících systémů, zařízení citlivých na čas, výrobních dílen a dalších nástrojů.
Tento systém byl vyvinut Victorem Yodaikenem v Novém Mexiku [WIKIPEDIE 2008].

Obrázek 5. Logo RTLinux

RTLinux je hard real-time (je deterministický) díky zlepšené kontrole správy přerušení mezi hardwarem a operačním systémem. Přerušení nutná pro deterministický chod jsou spravována real-time jádrem, zatímco ostatní přerušení jsou spravována Linuxem, který běží na nižší prioritě, než real-time vlákna. Pro kooperaci mezi real-time jádrem a operačním systémem může být použito sdílené paměti nebo zásobníku FIFO (typ datové fronty).

4.2.3. Windows CE

Jedná se o robustní embedded operační systém pro práci v reálném čase. Robustní jádro s podporou funkcí reálného času, nízkou latencí a deterministickým chováním.
Windows CE je optimalizován pro zařízení, která mají málo místa pro uložení operačního systému. Pro běh jádra systému Windows CE stačí velmi málo místa, v řádech jednotek megabajtů. Přístroje, na kterých se provozuje, jsou často bez zapisovatelných paměťových jednotek, a proto může být tento systém nakonfigurován jako takzvaný "uzavřený" systém, který už ale následně neumožňuje rozšíření (vše je uloženo v paměti ROM (typ paměťového čipu)). Windows CE je operační systém reálného času (RTOS). Podporuje 256 úrovní priority a používá přednostně dědičné priority pro práci s procesy [WIKIPEDIE 2008].



Obrázek 6. Windows CE.net

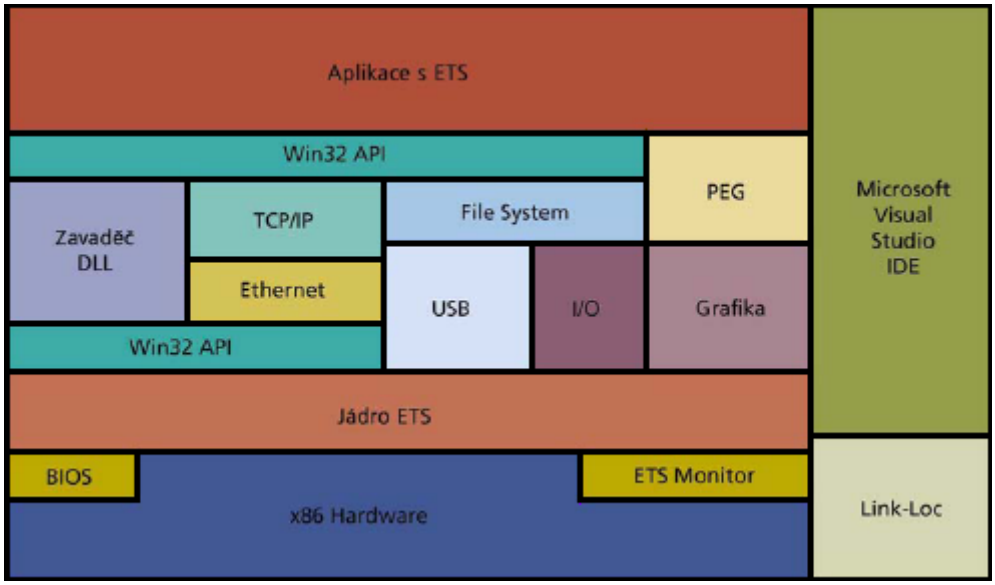
Mnoho ostatních operačních systémů je založeno na jádru tohoto systému (např. mobilní zařízení, digitální diáře (PDA), Pocket PC 2000, Pocket PC 2002, Windows Mobile 2003, Windows Mobile 2003 SE, 5,0 Windows Mobile, Windows Mobile 6, Smartphone 2002 a Smartphone 2003, digitální přijímače a přehrávače médií, slabé klienty se systémem Windows, průmyslové řídicí systémy, televizní adaptéry (Set-Top Box), telefony VoIP (Voice-Over IP), podnikové terminály a webové panely, brány, Microsoft AutoPC).

Ovladače pro hardware si obecně u tohoto systému vytvoří výrobce hardware sám. Společností Microsoft je dodáváno pouze uživatelské prostředí a vytvoření mezivrstvy mezi uživatelským prostředím a fyzickým hardwarem je již na výrobci tohoto hardware.

4.2.4. Phar Lap ETS

Phar Lap ETS je operační systém reálného času, který poskytuje programovým aplikacím vysoký výkon jejich provádění a který má optimální paměťový zábor, pouze o velikosti 88 kB. Vývojové prostředí ETS (SDK- knihovny a utility pro vývojáře) obsahuje sadu programátorských nástrojů, které jsou zcela integrovatelné do standardu - Microsoft Visual Studio IDE.
ETS používá stejné API (aplikační programové rozhraní) funkce, správu paměti, mutexy a semafory, které se používají pro Windows a to ho činní velice oblíbeným mezi programátory.

Architektura jádra ETS poskytuje širokou flexibilitu při implementaci, což umožňuje systémovým architektům využít jak monolitickou, tak dělenou implementaci jádro/aplikace. Jádro ETS je navrženo nad vysokorychlostním plánovačem, který využívá oba algoritmy: preemptivní i round-robin.
ETS podporuje neomezené množství programových vláken a zajišťuje jim jemné členění časové obsluhy ve 256ti úrovních přiřazení priorit vykonávání programových vláken. Plánovač zaručuje, že přepnutí kontextu kritického programového vlákna s vyšší prioritou se provede v časovém intervalu mezi 500 nanosekundami a méně než 2 mikrosekundami. Hodnoty časovače mohou být nastaveny v rozmezí 1 mikrosekunda až 18.2 milisekund [ARDENCE 2002 B].

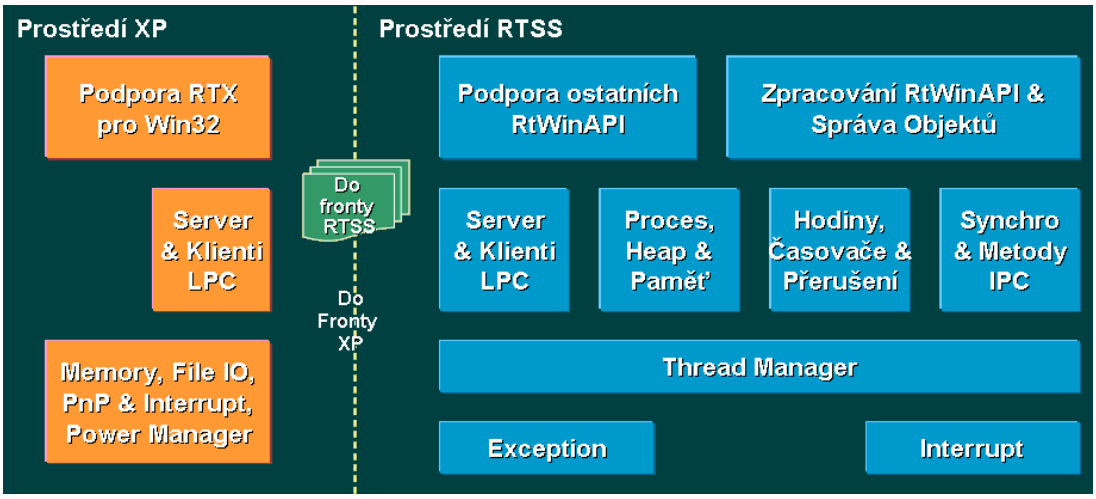


Obrázek 7: Architektura Phar Lap ETS [ARDENCE 2002 B]

5. IntervalZero RTX (Real-Time eXtension)

IntervalZero (dříve Ardence, dříve Citrix, dříve VenturCom) začal vyvíjet real-time produkty od roku 1980. Ze začátku jen na platformě UNIX.
OS Windows začal podporovat od roku 1995, když se stala populární platforma Windows NT 4.0.

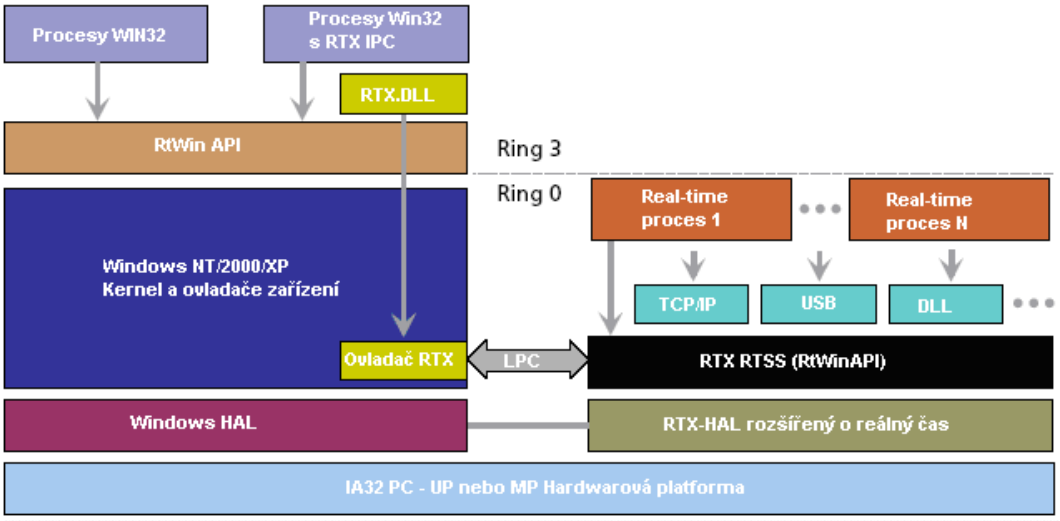
IntervalZero RTX je doplněk, který umožňuje deterministický běh a řízení aplikací v pevném reálném čase na standardní platformě Windows (NT/2000/XP a NT/XP embedded). RTX je těsně integrováno s jádrem Windows a pro svou činnost využívá služby Windows a rozhraní Win32 API. RTX je implementován jako kolekce knihoven (statických i dynamických), subsystém reálného času (RTSS) jako ovladač zařízení jádra Windows a rozšířený HAL (hardwarová abstrakční vrstva). Použití RTX eliminuje náklady na přídavný hardware a CPU pro reálný čas.



Obrázek 8: Detailní architektura propojení Windows XP a RTSS [ARDENCE 2002 A]

IntervalZero RTX se používá v aplikacích, které poskytují zvýšenou výkonnost, kontrolu a rozšiřitelnost, kombinovanou s provozní spolehlivostí, určených pro použití v průmyslové automatizaci, vojenství, letectví, testovacích a měřicích přístrojích, robotice a v mnoha dalších průmyslových odvětvích.
RTX není operační systém reálného času portovaný pod Windows, je to “jen“ subsystém reálného času. RTX zajišťuje precizní kontrolu nad IRQ (požadavek na přerušení), I/O a pamětí, takže se specifikované úkoly vykonávají se správnou prioritou a 100% spolehlivostí. Operacemi v Ring 0 zajišťuje RTX

nejvyšší výkon a vyžaduje minimální konfiguraci, s trvale podporovanou rychlostí obsluhy přerušení do 30 kHz a s průměrnou reakční dobou obsluhy přerušení kratší, než jedna mikrosekunda. RTX nepoužívá virtuální stránkování a to je jeden ze způsobů, kterými si RTX zajišťuje determinizmus.
RTX je rozšíření Windows, které využívá všechny standardní konvence Windows, včetně programátorského rozhraní API, správy paměti, SRI, mutexy a semafory. Aplikace s RTX může plně využít výhod mechanizmu ochrany paměti v Ring 3, kterou poskytuje Windows a architektura Intel.
Architektura RTX je rozšíření, které nijak nezasahuje do Windows, nepřekáží jejich běhu ani nemodifikuje žádnou z infrastruktury Windows. Díky tomuto oddělení subsystému reálného času RTX (real-time sub-systém = RTSS) je zajištěno, že aplikace běžící na bázi RTX přečkají i havárii OS Windows nebo systémové výjimky "blue screen".
Jádro RTX RTSS je navrženo nad vysokorychlostním plánovačem, který podporuje oba algoritmy "preemptivní" a "round-robin". RTX podporuje až 1000 nezávislých procesů, a v každém neomezený počet programových vláken.



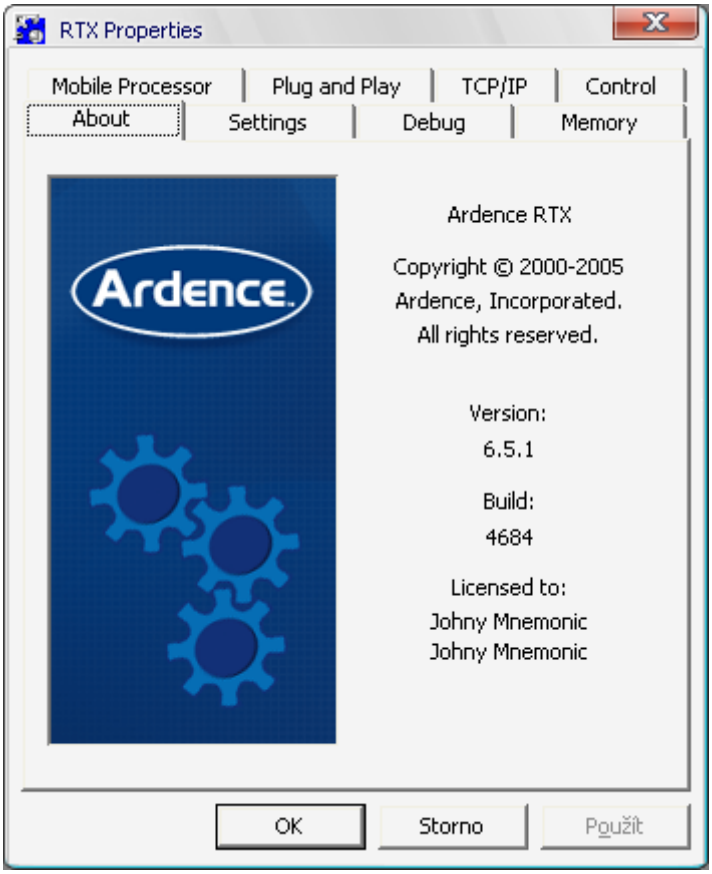
Obrázek 9: Architektura RTX [ARDENCE 2002 A]

Časově jemně členěné řízení aplikace zajišťuje obsluhu programových vláken až ve 256ti úrovní přiřazených priorit. Plánovač zaručuje, že přepnutí kontextu kritického programového vlákna a spotřeba času na přepnutí vlákna s vyšší prioritou se provede v časovém intervalu od 500 nanosekund do méně než 2 mikrosekund. Pro usnadnění komunikace mezi procesy RTX a aplikacemi Win32 při přenosu zpráv a synchronizaci mezi oběma prostředími, používá RTX

vysoce propustný mechanizmu s IPC (přenos dat mezi procesy). S použitím modelu sdílené paměti může IPC přesouvat velká množství dat bez ztráty výkonnosti.
Časově přesné vykonávání programu je v systémech reálného času kritické. Na podporu této přesnosti poskytuje RTX troje hodiny, které jsou základem časovačů událostí. Rozlišení hodin závisí na jejich použití, přesnost může být nastavena od 0.001 nanosekund, bez jakékoliv odchylky. Jsou podporovány intervaly časovačů 100, 200, 500 a 1000 mikrosekund. Počínaje reakční dobou IST (vlákno obsluhy přerušení), přes hodiny a časovače, plánovače, mechanizmy IPC až k nastavování priority vláken - každá komponenta RTX je optimalizována pro výkon.[ARDENCE 2002 A]

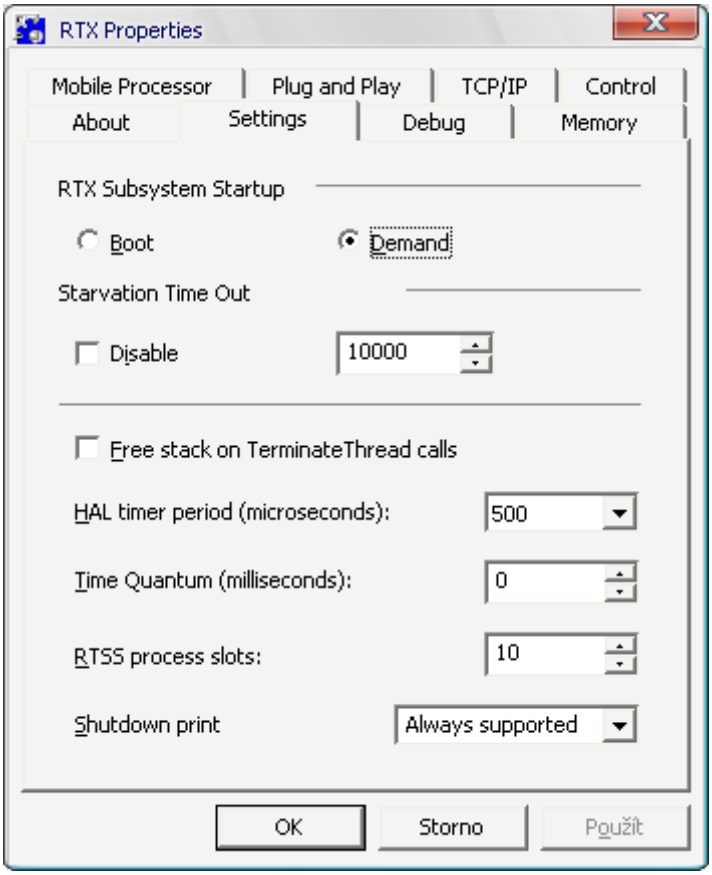
5.1. Nastavení vlastností RTX v operačním systému Windows:

Upřesnění základního nastavení doplňku RTX v systému MS Windows.



Obrázek 10: První záložka v nastavení vlastností RTX s popisem verze RTX

Na záložce About je popsána verze RTX, sestavení a licenční náležitosti. Případnou změnou na následující záložce Settings, můžeme výrazně ovlivnit chování a přesnost s jakou dokáže RTX pracovat. To se týká především nastavení HAL timer period, která je důležitá pro funkci časovačů v aplikacích RTX.



Obrázek 11: Nastavení časů reakcí doplňku RTX

Na záložce Settings se nastavuje kdy se má RTX v počítači spouštět. Může to být po spuštění počítače (Boot) nebo až přímým spuštěním uživatelem (Demand). Také se na ní nastavuje perioda HAL timeru a vyhrazený čas (pro práci aplikací v RTX).

Pro vývoj aplikací je kompletní sada nástrojů, které jsou zcela integrovány do prostředí Microsoft Visual Studio. V diplomové práci byly použity verze 6.5.1 a 8.1.1. Verze (6.5.1) nepodporuje zásuvné moduly pro Visual Studio 2005. A počáteční vývoj aplikací se proto prováděl ve Visual Studiu 6.0. Verze 8.1.1. již podporuje Visual Studio 2005 a proto byl vývoj částečně přenesen i do tohoto prostředí.

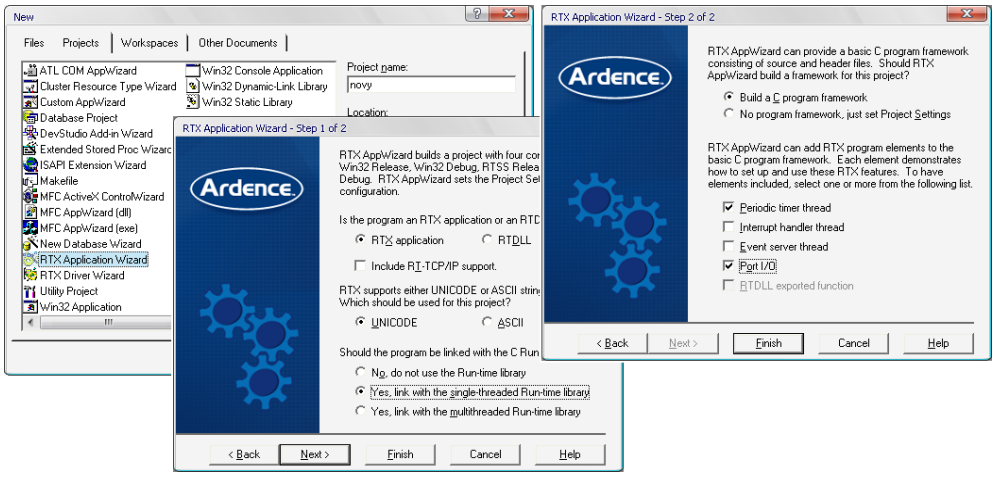
5.2. Vývojové nástroje RTX

Pro vytvoření projektu aplikace RTX v prostředí Visual Studia je k dispozici průvodce RTX Application Wizard. Jedná se o průvodce pro založení aplikace, umožňuje definovat konfiguraci RTX, která se hodí pro vyvíjenou aplikaci. Může se přesně specifikovat pracovní rámec (framework) pro aplikaci na bázi RTX.

Tento prvek ovšem musíme přidat do Visual Studia ručně (při použití Visual Studia 6.0 a verze doplňku RTX 6.5.1). Přidání provedeme následovně:

1. Zkopírovat *VenturcomDbgrs.dll* (obvykle naleznete v C:\Program Files\Ardence\RTX SDK\bin) do Program Files\Microsoft Visual Studio\Common\MSDev98,
2. ve Visual Studiu Tools menu, vyberte Customize,
3. zvolte Add-ins and Macro Files záložku a klikněte na Browse,
4. procházejte a hledejte *VenturcomDbgrs.dll* (obvykle najdete v C:\Program Files\Microsoft Visual Studio\Common\MSDev98),
5. označte Ardence Developer Studio Add-in a klikněte na Close.

Po provedení těchto úkonů se do nástrojové lišty přidají tři ikony RTX Debugger Properties, Display information's About RTX Debugger a RTX Help.



Obrázek 12: Application Wizard při tvorbě projektu

Začátek vytváření projektu RTX se nijak neliší od vytváření jakéhokoliv jiného projektu ve Visual Studiu. Zadáme název projektu a jeho umístění na disku. Po potvrzení se přesuneme na nové okno, toto okno je již průvodce nastavení projektu RTX.

V tomto okně volíme, jaký druh programu chceme vytvořit, volíme mezi dvěma možnostmi:

- RTX aplikací,
- RTDLL knihovnu.

Můžeme také přidat podporu RT – TCP/IP (podpora pro síťovou komunikaci).

Na tomto okně dále volíme formát programu:

- UNICODE,
- ASCII.

V dolní části okna zda má být program linkován pomocí run – time knihoven jazyka C. Zde máme na výběr ze tří možností:

- nepoužívat knihovny,
- používat, ale jen s jedním threadem,
- používat s multithreadem.

Nyní můžeme průvodce ukončit a dostat se tak přímo do vývojového prostředí Visual Studia potvrzením tlačítka Finish nebo můžeme pokračovat v nastavení projektu stisknutím tlačítka Next.

Po stisku tlačítka Next, se nám objeví poslední okno průvodce RTX pro založení projektu.

V tomto okně si volíme, zda mají být do programu přidány kusy základních kódů, které jsou již vytvořeny a programátorům mají ulehčit práci.

Při zvolení použití přidání kódů se nám zpřístupní dolní část okna a v ní volíme, které kusy kódu chceme do programu přidat:

- periodic timer thread (vlákno periodického časovače),
- interrupt handler thread (vlákno přerušovač),
- event server thread (událostní vlákno),
- port I/O (port pro vstup a výstup programu),
- RTDLL export function (exportní funkce pro RTDLL), (přístupná pouze, když vytváříme RTDLL knihovnu).

Projekt je v tomto okamžiku nastaven a potvrzením tlačítka Finish se přesuneme do vývojového prostředí Visual Studia.

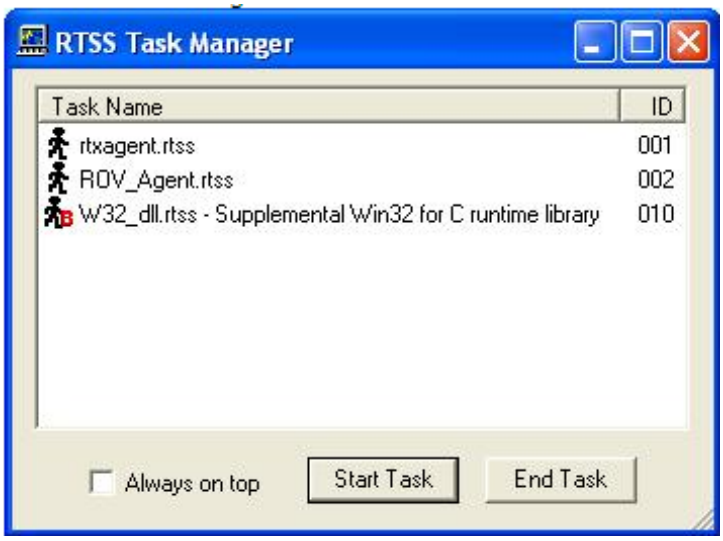
Tento postup je plně platný pro verzi RTX 6.5.1 Build 4684. Pro verzi RTX 8.1.1 Build 7180 je postup platný s drobnými obměnami.

5.3. Doplňky subsystému RTX

Subsystém reálného času nám nabízí takzvané pomocné programy pro testování a odlaďování námi vytvořených aplikací. Mezi tyto podpůrné programy patří:

5.3.1. RTSS Task Manager

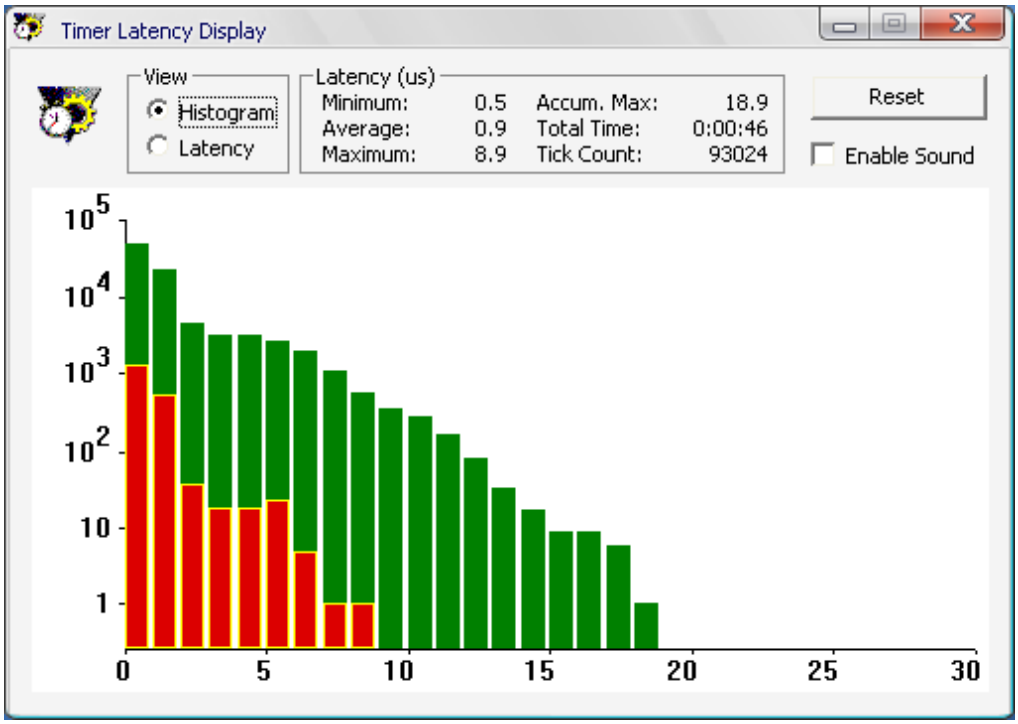
Aplikace zobrazuje interakce mezi procesy a použitými vlákny uvnitř RTX a aplikace na bázi RTX. Zobrazuje všechna přepnutí vláken, přepnutí kontextů. Časy vláken mohou být zobrazeny po kliknutí na jejich ID nebo časové značky.



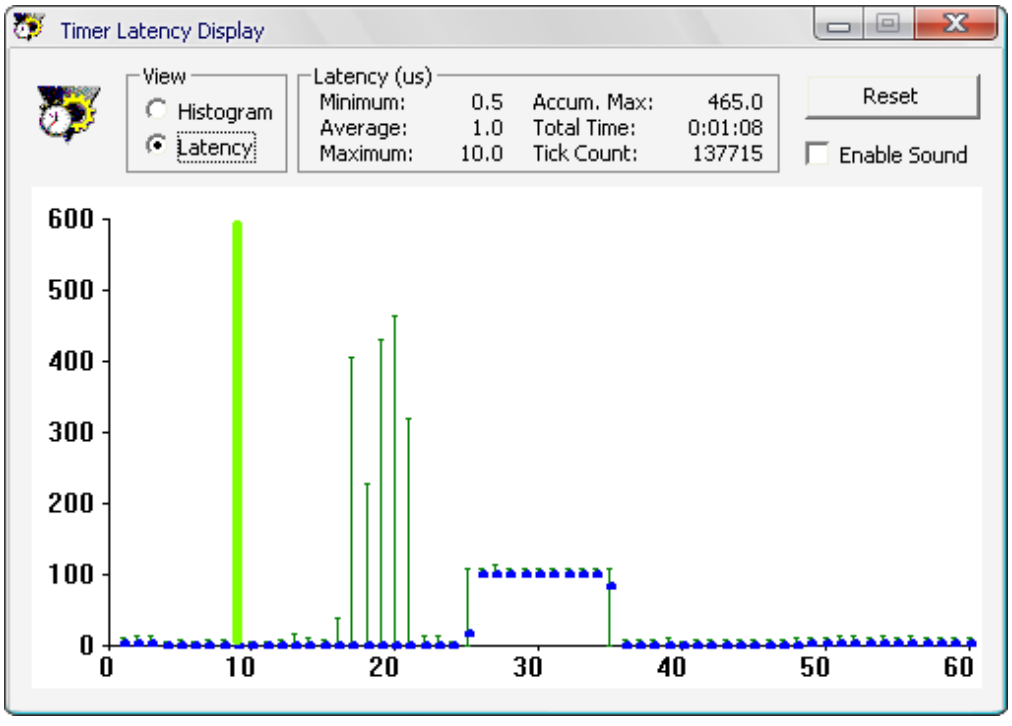
Obrázek 13. Okno aplikace Task Manageru

5.3.2. *Timer Latency Display*

Program Timer Latency Display zobrazuje reakční časy systému, které můžeme zobrazit ve formě histogramu nebo Latency.



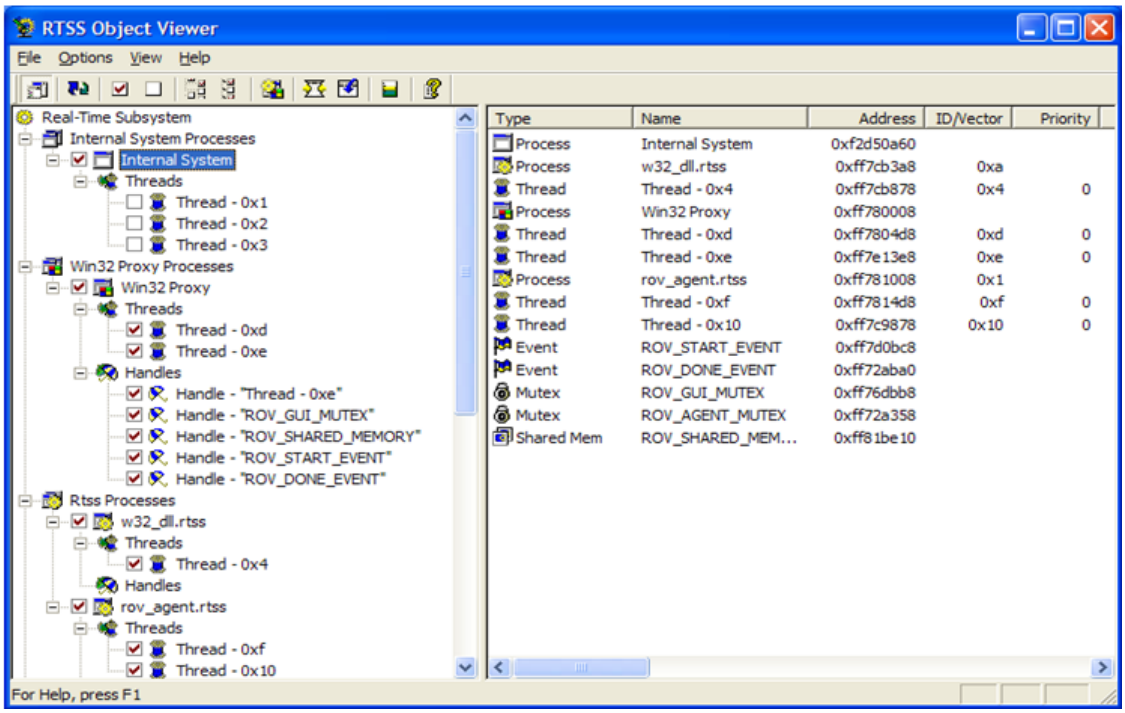
Obrázek 14: Timer Latency Display – Histogram typického zatížení OS Windows: práce disku, přehrávání videa, síťová komunikace, spořič obrazovky atd.



Obrázek 15: Timer Latency Display – Reakční časy z předchozího obrázku

5.3.3. RTSS Object Viewer

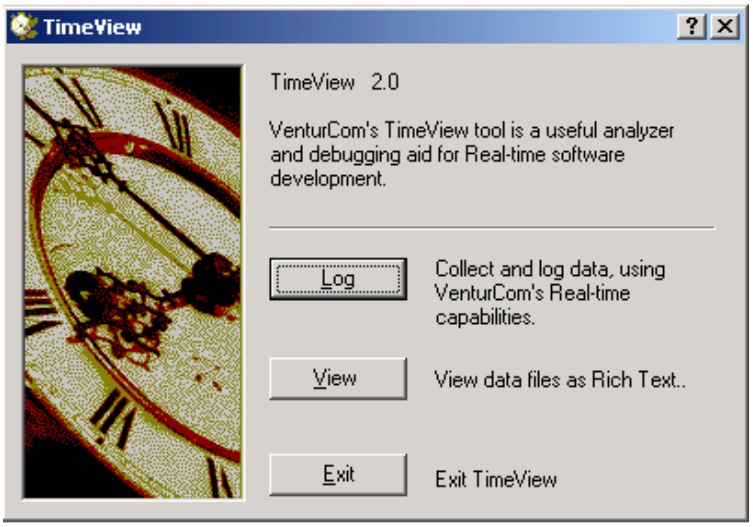
Umožňuje nám zobrazit přehled všech objektů, které běží v prostředí reálného času RTX, zobrazuje vlákna a globální procesy. Okno programu je rozděleno na dvě části, v levé je stromová struktura systému a v pravé detailní popis vybraného prvku.



Obrázek 16. Okno aplikace RTSS Object Viewer

5.3.4. TimeView

Program TimeView provádí zobrazení a konverzi log souborů mezi formátem Rich Text (.rtf) a WordPad.



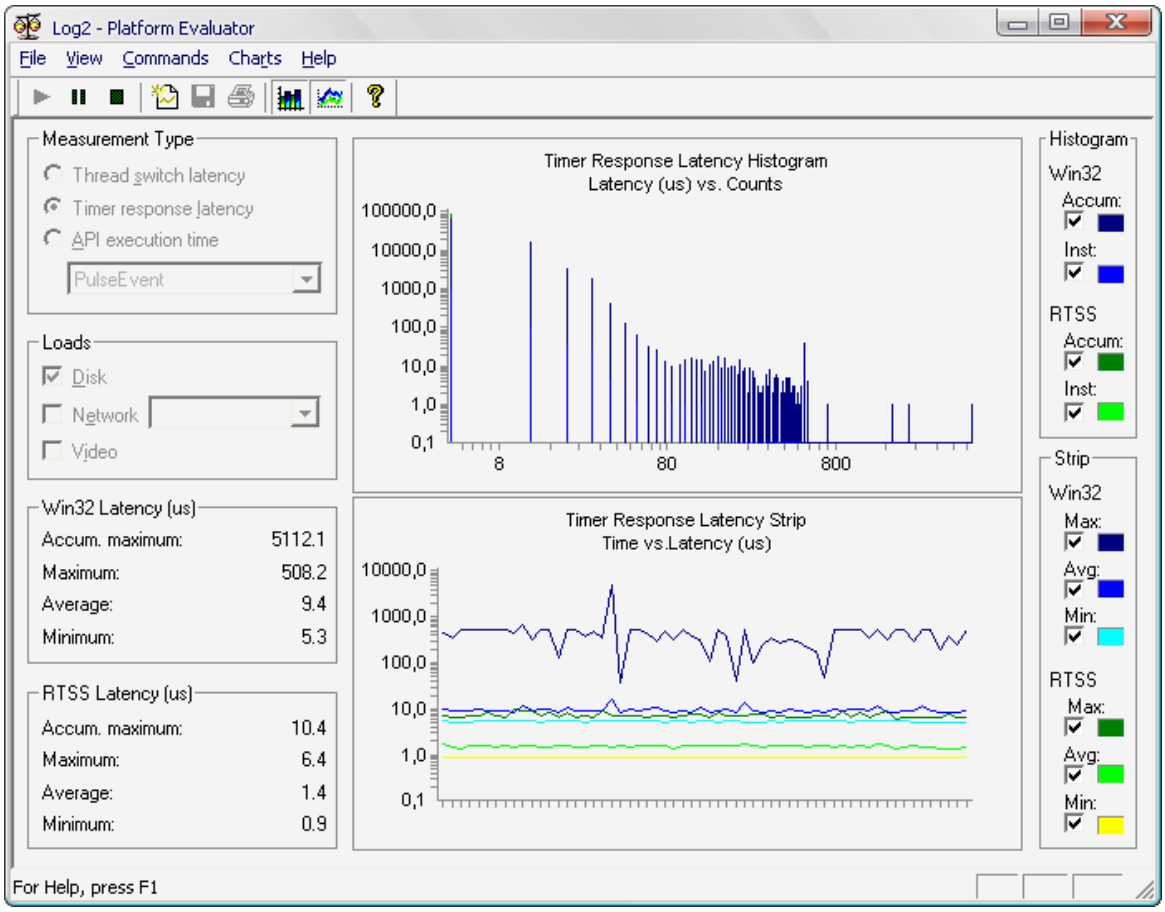
Obrázek 17. Okno program TimeView

5.3.5. *Platform Evaluator*

Tento vyhodnocovací nástroj výkonnosti reálného času pro Windows NT/ 2000/XP a NT/XP embedded. Umožňuje měnit zavádění systému, vybírat měřící kriteria a určovat přesné míry real-time kapacit na jakékoli platformě. Míry výkonu a s nimi spjaté konfigurační informace.

Měří a zobrazují časovací statistiky:

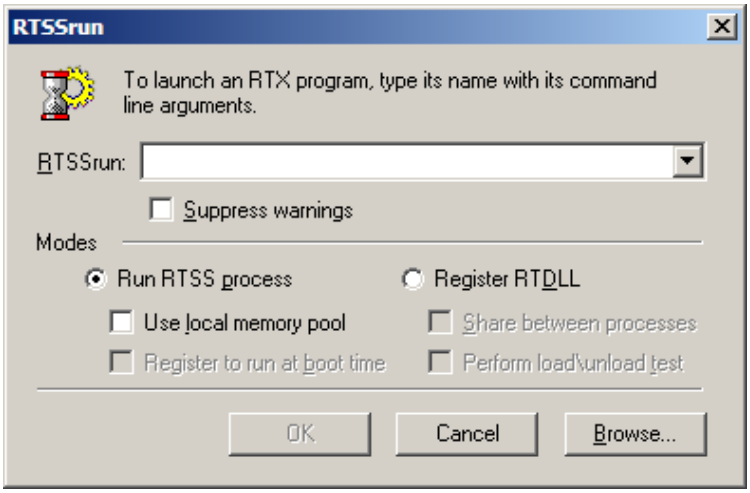
- doby odezvy,
- časy přepínání kontextu vláken,
- časy provádění kritických Win32 volání,
- vyhodnocuje výkon Win32 API,
- nabízí základní zavádění ovladačů pro vyhodnocení vhodnosti zařízení,
- graficky a numericky zobrazují výsledky výkonu,
- shromažďují informace o konfiguraci hardware a systému Windows.



Obrázek 18: Okno aplikace Platform Evaluator

5.3.6. RTSSrun

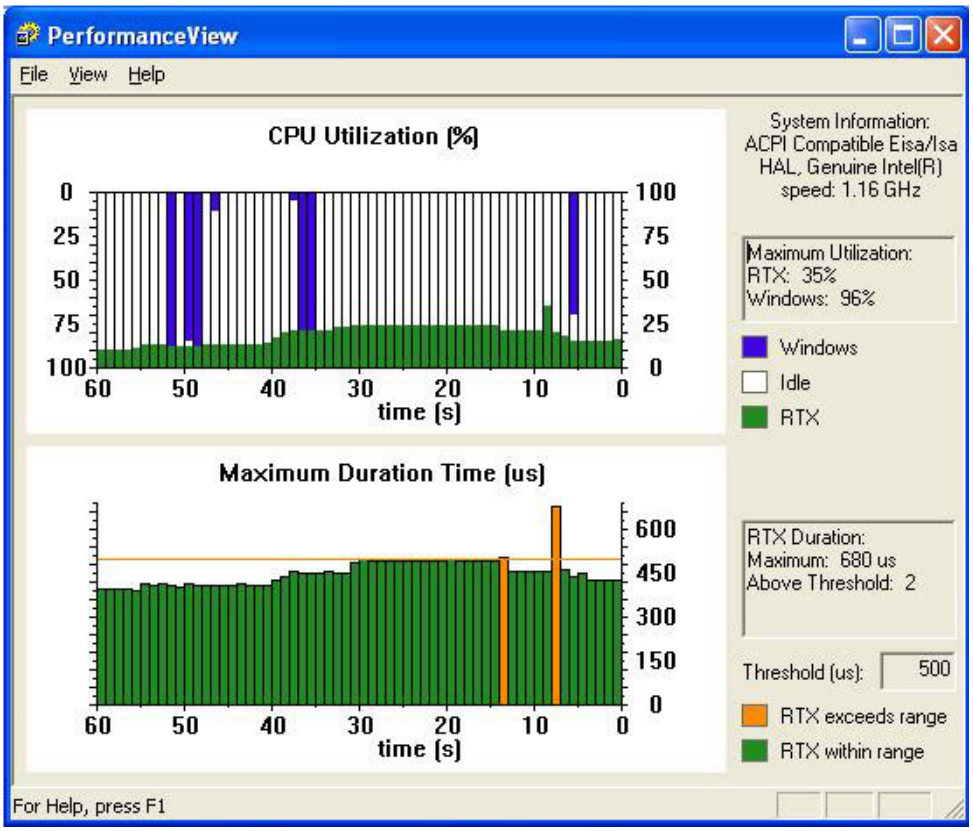
Jedná se o program, který spouští RTSS procesy a registruje DLL (dynamicky vázaná knihovna).



Obrázek 19. Okno programu RTSSrun

5.3.7. PerformanceView

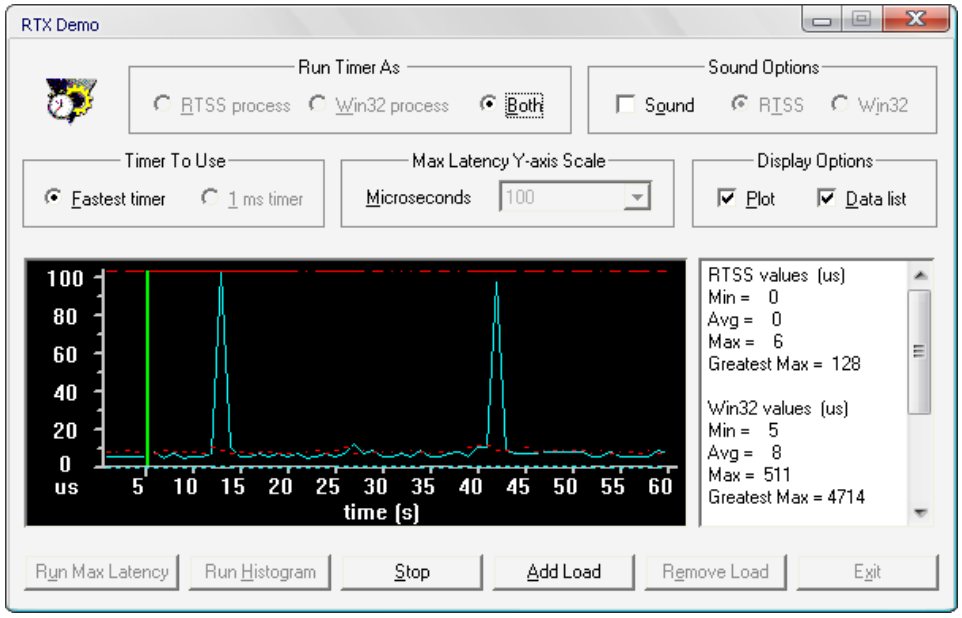
Tento program monitoruje využití CPU, jak ze strany Windows, tak RTX a zvyšuje pro vývojáře aplikací RTX přehled o využití CPU.



Obrázek 20. Okno aplikace PerformanceView

5.3.8. RTX Demo

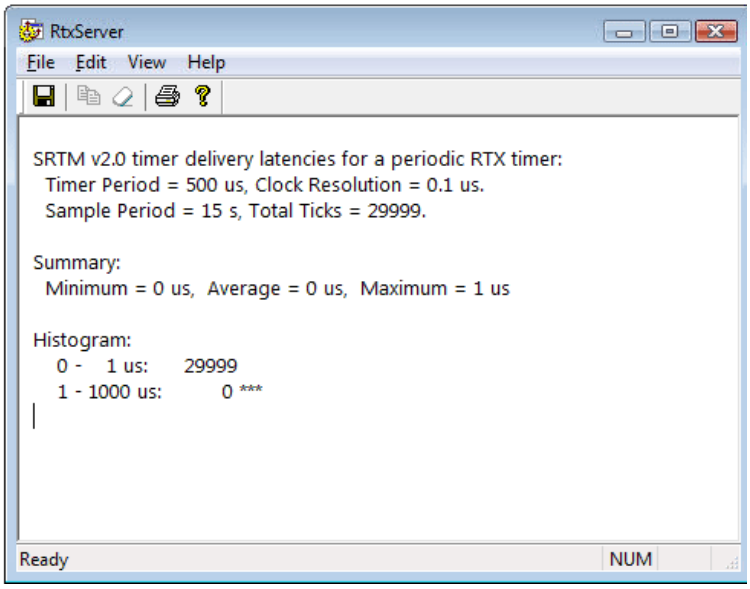
Program RTX Demo je demonstrativní program, zobrazující délku časů do splnění požadavků v systémech Windows a RTSS.



Obrázek 21: RTX Demo, srovnává RTSS a Win32 časy požadavků do splnění

5.3.9. RTX Server Console

Zjednodušeně se dá říci, že RTX Server je obdoba příkazového řádku, se kterým se můžeme setkat v prostředí MS Windows. Tento server ale funguje jen jedním směrem, to znamená, že data pouze vypisuje, poskytuje nám jednoduchý výpis událostí, které jsou výstupními informacemi (print funkce) z běžících programů.



Obrázek 22. Okno RTX Serveru

5.4. Tvorba ovladačů prostřednictvím technologie RTX

Na rozdíl od Windows Driver Model (WDM - rámec pro ovladače), přerušení pod RTX běží v plném kontextu vláken programu a má k dispozici kompletní API, včetně funkcí plovoucí řádové čárky. Využití modelu programových vláken pro ovladače zařízení nabízí programátorům velkou flexibilitu v nastavování priorit a rozdělování funkcí mezi aplikace a ovladače. RTX poskytuje flexibilní IPC mechanizmus na propojení aplikací a ovladačů. K dispozici je rovněž mnohem výkonnější mezivrstva DCX (Data Control and Exchange), která poskytuje standardní, snadno použitelné rozhraní, tok dat a propojitelnosti mezi ovladačem zařízení a součástmi programu. Ovladače a programy jsou vzájemně nezávislé a zaměnitelné [ARDENCE 2002 B].

5.5. Vytvoření Real-Time (RT) programu

Pro vytvoření RT programu (*.rtss) se používá IDE Visual Studia (.NET).
Lze vytvořit programy dvou typů *.exe je spustitelný ve Win32 a *.rtss, který je spustitelný v RTSS. Programy s koncovkou exe běží pod systémem MS Windows a nejsou deterministické, přesto že využívají některé deterministické funkce RTX. Funkce používané pro vytvoření kódu kompilovatelného do prostředí RTX se v podstatě, ve svém grafickém tvaru, odlišují od standardních funkcí v prostředí MS Windows, předponou Rt (např. v prostředí MS Windows se používá funkce “printf“ a v prostředí RTX se místo ní používá “RtPrintf”).
Při programování aplikace vytváříme pointery (ukazatele) na adresy v paměti v nestránkovaném pool-u. V tomto případě mohou být používány pointery s referencí do Win32 i RTX a tímto způsobem se předávají objekty mezi *.exe a *.rtss. RTX zajišťuje synchronizaci objektů Mutexy, semaphory, eventy a lze signalizovat událost mezi *.exe a *.rtss.

5.6. Přidání zařízení pod správu RTX

Přidání zařízení pod správu RTX je nedílná podmínka správného fungování RTX aplikace. Pokud nedojde k přidání zařízení pod správu RTX, tak riskujeme nedeterminismus, pád či nespolehlivost fungování aplikace při výjimkách v systému windows, jako jsou např. BSOD.

Přidání zařízení je detailně popsáno v Příloha B, kde je i popsáno nastavování zařízení pod správou RTX (Příloha C). V Příloha D je také popsáno i zpětné převedení zařízení pod opětovnou správu MS Windows.

Poznámka autora:

Ne vždy však tato operace funguje korektně. Například se vyskytly problémy při práci s multifunkční kartou PCA – 7228AS.

*Tato chyba se vyskytla jen na jednom počítači s operačním systémem Windows XP Professional. V případě převádění zařízení (měřicí karty PCA – 7228 AS), ve školních laboratořích na počítači s operačním systémem Windows 2000, se tato chyba nevyskytla, jen bylo nutné mezi jednotlivými převody počítač restartovat. Popis chyby je následující: Přidání karty PCA – 7228 AS pod správu RTX proběhne jen, pokud nenainstalujeme originální ovladače měřicí karty. Po přidání měřicí karty pod správu RTX, se karta zobrazí jako „**Rtx PnP and Power Management Device**“. Po nainstalování originálních ovladačů měřicí karty již není možné měřicí kartu přidat pod správu RTX a v možnostech výběru ovladače karty se zobrazí jen originální ovladač od společnosti Tedia, ovladač společnosti Ardence již není dostupný a to ani při přímém zvolení ovladače Ardence.*

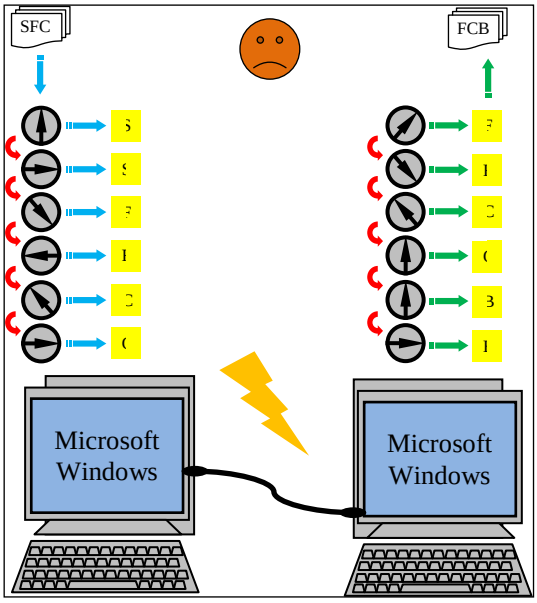
6. Úloha demonstrující schopnosti doplňku RTX

Pro demonstraci byla zvolena komunikace přes standardní paralelní port (LPT) mezi dvěma počítači. Úloha je založena na časové synchronizaci, kdy jeden počítač v určitý okamžik vyšle data na LPT port a v ten samý okamžik druhý počítač čte z LPT. Jednotlivé odesílání a čtení musí probíhat v přesně stanovený čas (periodu), aby byla přečtena z LPT portu správná data (viz Obrázek 23 a 24).

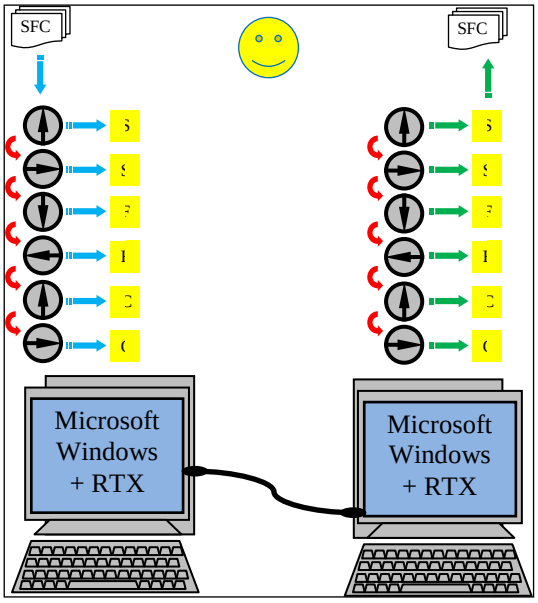
Postup demonstrace je následný: propojení dvou počítačů pomocí paralelního portu a přes toto propojení následně přenášet data.

LPT lze použít k přímému spojení počítačů bez nutnosti osazení síťové karty a instalování speciálního síťového softwaru. Při propojení dvou počítačů lze použít pro přenos pouze 5 datových kabelů, neboť máme k dispozici pouze 5 vstupních (stavových) signálů.

Standardní propojení paralelních portů, se provádí pomocí tzv. Laplink kabelu.



Obrázek 23. Časová disharmonie při přenosu dat, výsledkem je nesprávné přenesení dat



Obrázek 24. Dodržení periodického vysílání a čtení z LPT, výsledkem je správné přenesení dat

V operačním systému Microsoft Windows NT (ale i Windows 2000 nebo XP) nelze k registrům paralelního portu (LPT) přistupovat přímo, a proto byl na katedře ATŘ vytvořen speciální systémový program (ovladač zařízení – device driver), který LPT zpřístupní. Tento ovladač je nutné užít pouze, chceme-li přistoupit na paralelní port v prostředí MS Windows, nevyužívaje RTX.

Protože standardní operační systémy nám nedovolují přistupovat přímo k hardware počítače, ale tento přístup nám zprostředkovávají operační systémy přes různé vlastní ovladače, musíme použít výše zmíněný ovladač.
Tento problém pod systémem RTX nevzniká, jelikož máme možnost plného a přímého přístupu k hardware počítače. To znamená, přímou komunikaci mezi aplikací a hardwarem (mezi komunikací již není přítomná žádná další vrstva, jež by mohla mít za následek prodloužení doby komunikace a možného znehodnocení přenášených údajů). Toto je jedna ze základních důležitých vlastností systému RTX, který z něj dělá operační systém reálného času.

Při dodržení vlastností operačního systému reálného času by tento přenos dat měl být bezztrátový. Ale jak se dočteme dále, v prostředí MS Windows dochází, při zmenšení časové prodlevy a zatížení systému, ve vysílání a přijímání dat k časovému posunu. A jak přijímaní tak vysílání je v disharmonii a dochází ke značnému znehodnocení a ke ztrátám. Ale v případě systému RTX tomu tak není, tam vše probíhá i ve stavu vytížení počítačů téměř se stoprocentní přesností. A tímto tato jednoduchá úloha jednoduše demonstruje schopnosti systému RTX.
Program pro přenos dat je jak pro prostředí Windows, tak pro prostředí RTX vytvořen ve Visual Studiu 2005 v jazyku C/C++ (verze RTX 6.5.1 neobsahuje zásuvné moduly pro tvorbu projektu a ovladačů pro Visual Studio 2005, ale novější verze 8.1.1 již ano). Operační systém testovaných počítačů byl MS Windows 2000 SP4.

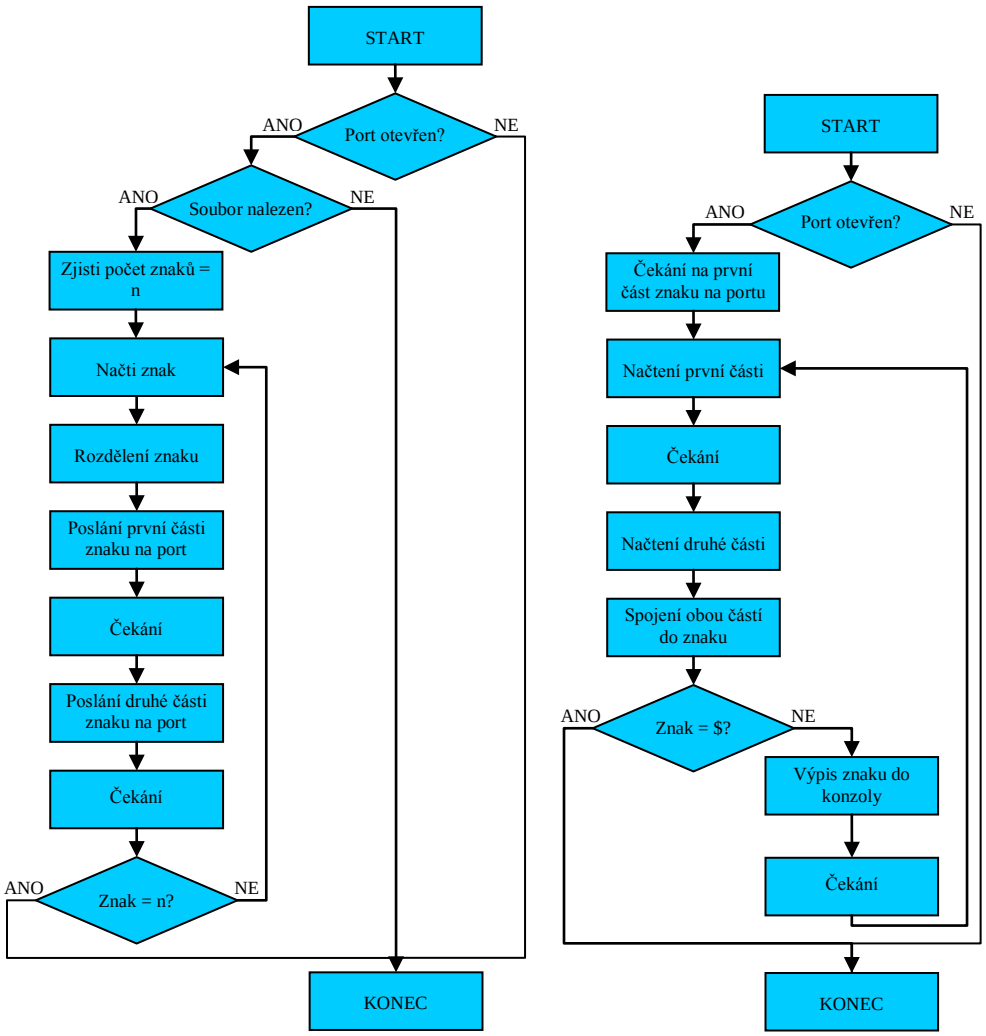
Program vysílá data z počítače č. 1 (vysílače), na počítač č. 2 (přijímač). Vysílání a přijímání dat na paralelním portu, je prováděno v časovém intervalu. Vysílač zahájí vysílání dat a mezi jednotlivým vysíláním přerušuje činnost na daný časový úsek (byly zvoleny intervaly 100, 50, 25, 10 a 1 milisekund). Přijímač přijetím první části znaku také přerušuje svojí činnost, a to na dobu stejnou, jaká je nastavena na vysílači.

Programy fungují na principu rozdělování/spojování, vysílaného/načteného znaku na dvě poloviny (přes LPT port nelze přijmout celý jeden znak najednou, více viz Příloha A) a následném odesílání první a druhé poloviny a to v daných časových intervalech. Po odeslání posledního znaku z načteného souboru

je vyslán poslední tzv. ukončovací znak (v našem případě byl zvolen symbol dolaru $), po jeho odeslání/přijetí programy ukončí svoji činnost.

Algoritmus vysílacího programu je následovný. Program otevře zdrojový soubor s daty a zjistí počet znaků v souboru. Poté začne vysílat data na paralelní port a mezi jednotlivým vysíláním pozastavuje svoji činnost na předem nastavený časový interval. Po odeslání všech znaků ze zdrojového souboru vyšle ukončovací znak a ukončí svoji činnost.
Algoritmus přijímacího programu na počítači funguje následovně. Po spuštění čeká na načtení prvního znaku z paralelního portu a poté načítá v nastaveném intervalu data z paralelního portu a zobrazuje je na obrazovce. Po přečtení ukončovacího znaku program ukončí svoji činnost.

Vývojové diagramy komunikace v prostředí MS Windows



Vývojový diagram 1 a 2: Programy vysílající a přijímací znaky z paralelního portu (vysílač v levo, přijímač v pravo)

Programy byly testovány v prostředí MS Windows a to ve dvou stavech, na nezatíženém a zatíženém počítači. Nejprve bez užití doplňku reálného času RTX a následně v doplňku reálného času RTX.

6.1. Přenos dat přes paralelní port (LPT) v prostředí MS Windows

V průběhu přenosu dat v systému windows dochází, při vysokém vytížení počítače k prodloužení časů na zpracování požadavků na CPU. V tomto důsledku dochází k nekorektnímu běhu programu (program odesílá informace na paralelní port se zpožděním). V tomto důsledku dochází k nesprávnému čtení dat na přijímacím počítači. Tato situace je analogická a s podobným výsledkem, jako když vytížíme přijímací počítač při čtení dat z paralelního portu. Také v tomto případě dochází k prodloužení intervalů či dýpadkům a tím pádem ke špatnému čtení dat z LPT.
Pro přenos byl vytvořen textový soubor, který obsahuje posloupnost čísel od 1 do 50. Na jednom řádku je vždy jedno číslo následované čárkou.
Jako časovače, který určuje prodlevu, mezi čtením byla zvolena funkce Sleep(), jež je standardní funkcí jazyka C/C++ pro dočasné „uspání“ procesu.
Tato funkce, ale nezaručuje dostatečnou přesnost, jelikož je nedeterministická. Pokud použijeme tuto nedeterministickou funkci i v prostředí RTX, tak dochází ke ztrátám dat také (pro zamezení je nutné používat deterministickou funkci RtSleepEx).

6.1.1. Přenos dat na nezatíženém počítači

Oba počítače jsou nezatížené (na obou je spuštěno pouze přenášení souboru přes paralelní port). Data, která přenášíme z vysílače na přijímač, jsou co do velikosti objemu dat i obsahu z 33% stejná. Celková úspěšnost přenosu dat se pohybuje okolo 33%. Tabulka porovnání naměřených hodnot a grafu úspěšnosti jsou uvedeny na konci kapitoly.

6.1.2. Přenos dat na zatíženém počítači

Při přenášení dat z vysílače je na přijímači spuštěno přehrávání videa (popřípadě kopírování souboru z DVD na pevný disk). Při tomto zatížení je z přenesených dat (vysílaný z vysílače na přijímač přes paralelní port) zřejmé

po dokončení přenosu (ve většině případů se přijímací program musí ukončit manuálně, jelikož program správně nepřečte ukončovací znak a stále se pokouší načítat data z paralelního portu), a jeho následném otevření, že přenos neproběhl správně a přenesená data jsou výrazně poškozena (velikost dat je jiná a obsah se neshoduje se daty vysílanými ze serveru). Celková úspěšnost přenosu dat se pohybuje okolo 15%. Tabulka porovnání naměřených hodnot a grafu úspěšnosti jsou uvedeny na konci kapitoly.

6.2. Přenos dat přes LPT v prostředí RTX

Problém nedeterministických funkcí byl v prostředí RTX vyřešen použitím deterministických časovačů z prostředí RTX (tyto funkce samozřejmě nelze aplikovat na programy běžící v prostředí Windows).
V průběhu přenosu dat v subsystému RTX nedochází při vysokém vytížení počítače, k prodloužení časů na zpracování požadavků na CPU. V tomto důsledku nedochází k nekorektnímu běhu programu (program odesílá data na paralelní port bez zpoždění). V tomto důsledku dochází ke správnému čtení dat na přijímacím počítači. Tato situace je analogická a s podobným výsledkem, jako když vytížíme při přijímání dat z paralelního portu přijímací počítač.
Také v tomto případě nedochází k prodloužení časů a tím pádem ke správným údajům čteným z LPT. Při nízkých rychlostech dochází k ojedinělým výpadkům, ale tyto výpadky jsou zapříčiněny samotným hardwarem.
V tomto případě je použito jak standardních funkcí jazyka C a C++ tak také funkcí z prostředí RTX. Zdrojové soubory, ze kterých byl vygenerován *.rtss soubor, se liší jen v detailech (několik funkcí je v tomto případě zaměněno z jazyka C a C++ deterministickými funkcemi z prostředí RTX).

6.2.1. Přenos dat na nezatíženém počítači

Při tomto přenosu dat z vysílacího počítače na přijímací počítač, jsou oba počítače nezatížené a je na nich spuštěna pouze úloha přenosu dat přes paralelní port. V tomto případě proběhne přenos dat téměř v pořádku a přenesená data jsou zcela shodná jak co do velikosti, tak co se obsahu týče. Procentuální úspěšnost přenosu je ale podstatně vyšší než v případě MS Windows prostředí. Úspěšnost přenosu dat se pohybuje okolo 100%.

Tabulka porovnání naměřených hodnot a graf úspěšnosti jsou uvedeny na konci kapitoly.

6.2.2. Přenos dat na zatíženém počítači

Při tomto přenosu dat z vysílacího na přijímací počítač, je přijímací počítač během přenosu zatížen spuštěním video souboru (nebo např. kopírováním souborů z DVD atd.). Při dokončení přenosu dat se při následném ověřování můžeme utvrdit v tom, že úspěšnost je opět výrazně vyšší než v případě prostředí MS Windows a pohybuje se kolem 99,9%.
Tabulka porovnání naměřených hodnot a úspěšnosti je na konci kapitoly.

6.3. Dílčí výsledky demonstrační úlohy

Tato demonstrace dokazuje, že v případě použití doplňku RTX s užitím deterministických funkcí, dokáže RTOS v operačním systému Windows zajistit, že procesy, o které se stará, dokážou pracovat zcela nezávisle na stavu operačního systému a jeho momentálním zatížení. V tabulkách níže je vidět, že i pod systémem RTX dochází ke ztrátám, k těm ale v tomto případě dochází né kvůli chybě v RTX, ale v důsledku hardware, který při nízkých rychlostech není schopen pracovat vždy zcela spolehlivě.

Pro systém RTX musely být vytvořeny časovače pro zajištění přesnosti synchronizace přenosu dat, a to následovně:
Nejprve byly vytvořeny hodiny:

```
// nastaveni delky „periody“ a nasledne vytvoreni vlakna hodin
liPeriod.QuadPart = 5000;
if(!(hTimer = RtCreateTimer(NULL,              // bezpecnost
                            0,                 // 0 = default
                            TimerHandler,      // casove vlakno
                            NULL,              // NULL argument na
                                                  vlakno
                            RT_PRIORITY_MAX,   // nastaveni max.
                                                  priority
                            CLOCK_FASTEST))){ // RTX HAL timer
      RtWprintf(L"RtCreateTimer error = %d\n",GetLastError());
      ExitProcess(1);   // v pripade neuspechu vypise chybu a skonci
}
```

a následně byla vytvořena periodická funkce:

```
// vytvoreni periodickeho opakovani (hodiny, do zacatku, perioda)
if(!RtSetTimerRelative(hTimer, &liPeriod, &liPeriod)){
      RtWprintf(L"RtSetTimerRelative error = %d\n",GetLastError());
      RtCancelTimer(hTimer, NULL);// v pripade neuspechu vypise chybu
      ExitProcess(1);              // a ukonci se
```

```
}
```

Popis úpravy bitů pro zápis a čtení na paralelním portu viz Příloha A.

Zdrojové sobory komunikačních programů viz Příloha E.

Tabulka 1: Přenos dat pod systémem Windows bez RTX (nezatížený systém)

	Windows								
	pokus	1	2	3	4	5	6	průměr	%
časová prodleva mezi odesíláním	100 ms	140	139	138	28	89	65	99,83	71,31
	50 ms	140	47	6	138	139	108	96,33	68,81
	25 ms	3	3	3	3	8	3	3,83	2,74
	10 ms	1	3	90	8	91	8	33,50	23,93
	1 ms	5	1	6	4	1	1	3,00	2,14
		správně přenesených znaků						Celkem	

Tabulka 2: Přenos dat pod systémem Windows bez RTX (zatížený systém)

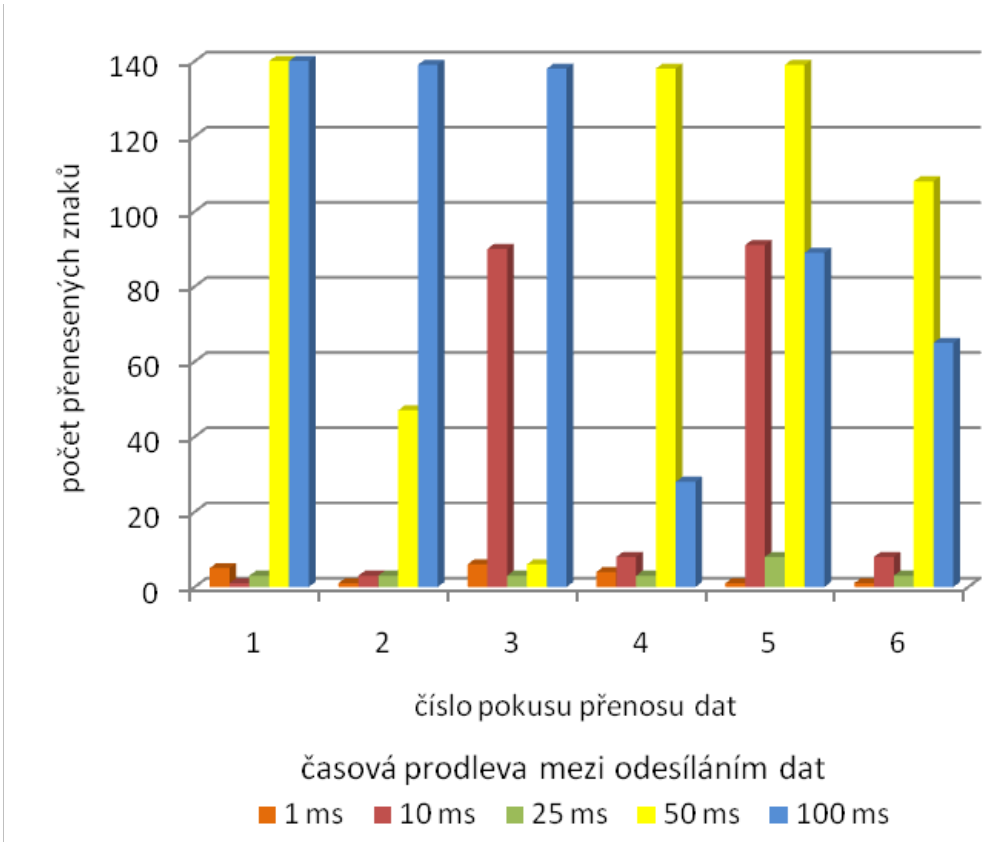
	Windows								
	pokus	1	2	3	4	5	6	průměr	%
časová prodleva mezi odesíláním	100 ms	30	45	26	67	77	65	51,67	36,90
	50 ms	40	47	6	13	23	27	26,00	18,57
	25 ms	3	50	3	12	18	3	14,83	10,60
	10 ms	1	3	15	8	40	8	12,50	8,93
	1 ms	0	1	2	3	1	1	1,33	0,95
		správně přenesených znaků						Celkem	

Tabulka 3: Přenos dat pod systémem Windows s RTX (nezatížený systém)

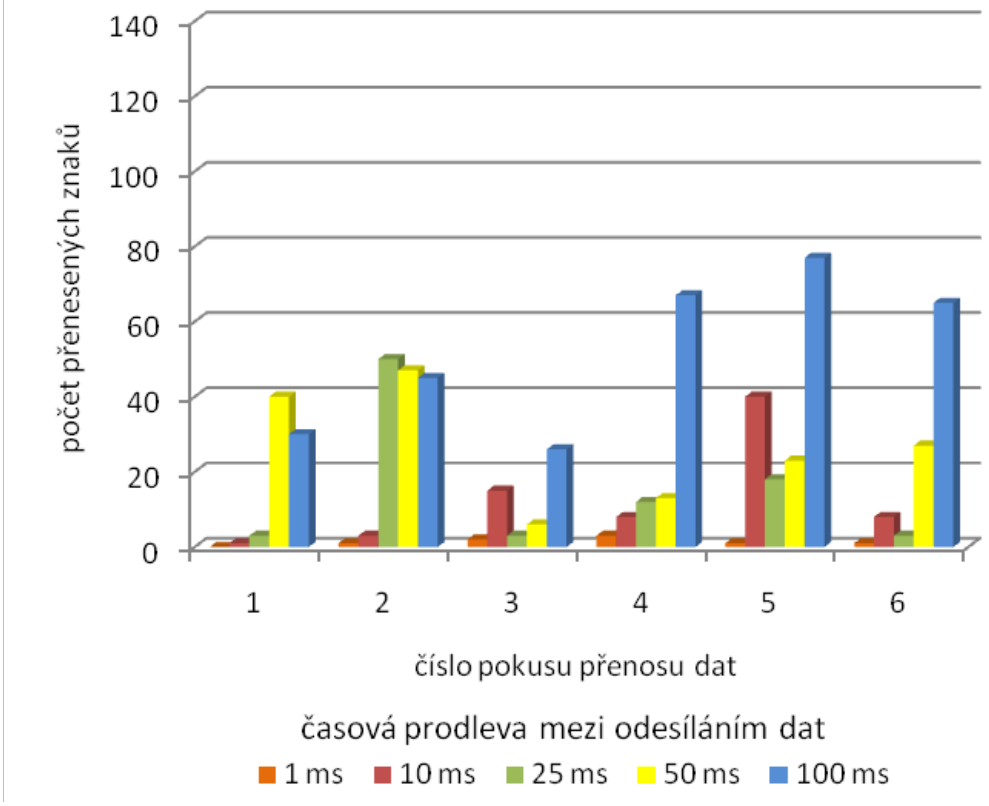
	Windows + RTX								
	pokus	1	2	3	4	5	6	průměr	%
časová prodleva mezi odesíláním	100 ms	140	140	140	140	140	140	140,00	100,00
	50 ms	140	140	140	140	140	140	140,00	100,00
	25 ms	140	140	140	140	140	140	140,00	100,00
	10 ms	140	140	140	140	140	140	140,00	100,00
	1 ms	140	139	140	140	140	140	139,83	99,88
		správně přenesených znaků						Celkem	

Tabulka 4: Přenos dat pod systémem Windows s RTX (zatížený systém)

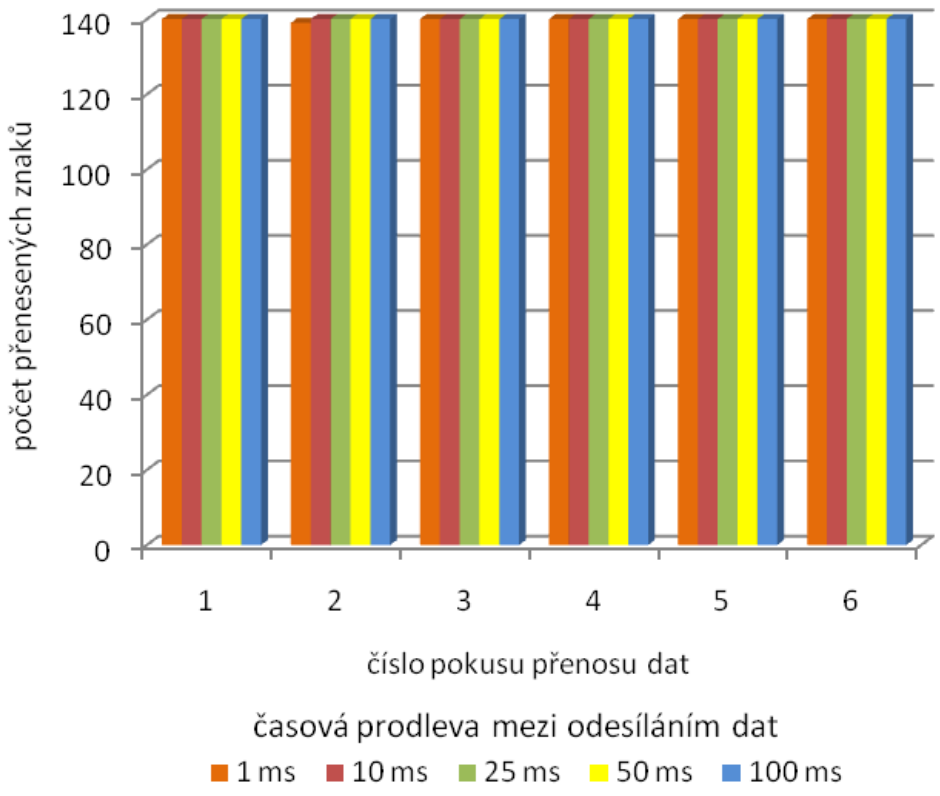
	Windows + RTX								
	pokus	1	2	3	4	5	6	průměr	%
časová prodleva mezi odesíláním	100 ms	140	140	140	140	140	140	140,00	100,00
	50 ms	140	140	140	140	140	140	140,00	100,00
	25 ms	140	140	140	140	140	140	140,00	100,00
	10 ms	140	139	140	140	140	140	139,83	99,88
	1 ms	140	140	140	139	140	140	139,83	99,88
		správně přenesených znaků						Celkem	



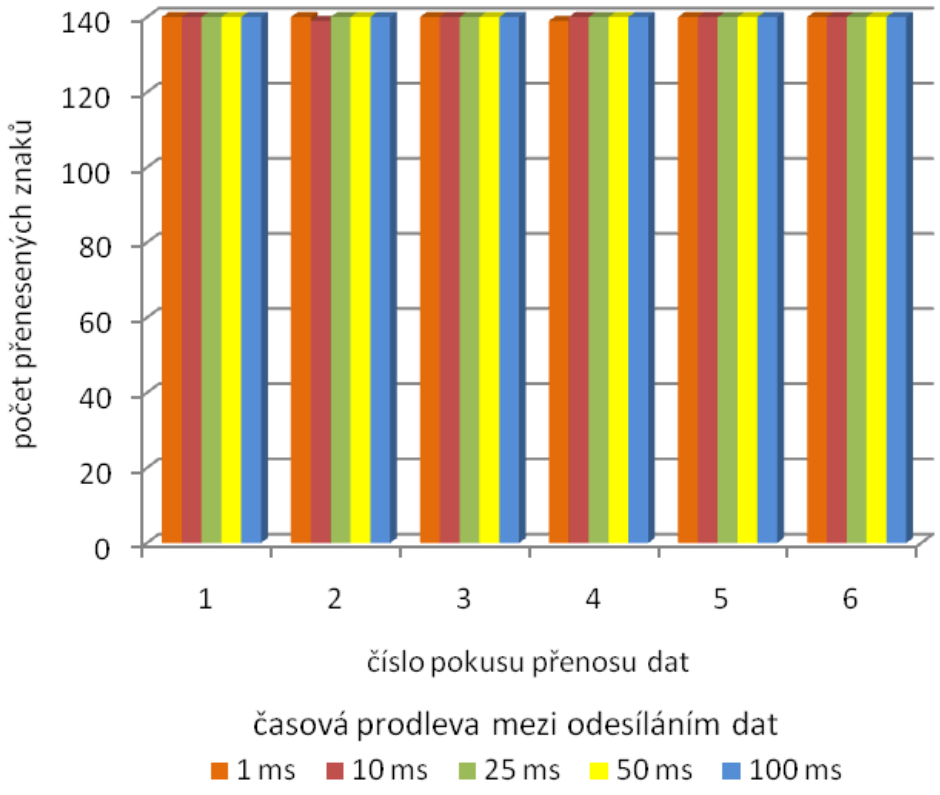
Graf 1: Úspěšnost přenosu dat pod systémem Windows bez RTX (nezatížený systém)



Graf 2: Úspěšnost přenosu dat pod systémem Windows bez RTX (zatížený systém)



Graf 3: Úspěšnost přenosu dat pod systémem Windows s RTX (nezatížený systém)



Graf 4: Úspěšnost přenosu dat pod systémem Windows s RTX (zatížený systém)

7. Multifunkční měřicí karty

Multifunkční měřicí karty neboli karty pro sběr dat (DAQ), používáme pro přímé měření nebo generování signálu počítačem. Tímto použitím se liší např. od karet realizujících rozhraní GPIB, které pouze komunikují se specializovaným měřicím přístrojem.

Standardní multifunkční karta obsahuje:

- analogové vstupy,
- analogové výstupy,
- digitální linky,
- čítače anebo časovače.

Záleží na kartě, kolik z předchozích položek obsahuje.

Podstatnou vlastností zejména pro on-line zpracování dat je schopnost nastavit práh zaplnění bufferu pro vyvolání přerušení (IRQ). Karty osazené standardní FIFO zásobníkem, umožňují vyvolat přerušení při zaplnění 50% kapacity (což je výhodné pro nejvyšší vzorkovací frekvence) nebo při zaplnění alespoň jedním naměřeným vzorkem (vhodné pouze pro nejnižší frekvence).

7.1. Multifunkční měřící PC karta PCA 7228 AS

Karta PCA – 7228 AS představuje výkonnou multifunkční PC kartu pro univerzální použití nabízející všechny obvyklé I/O funkce.
Karta obsahuje osm analogových vstupů (s možností rozšíření na 32 pomocí externího multiplexoru OPT-832 nebo OPT-830B) alternativně s 12bitovým nebo 14bitovým A/D převodníkem a velkým datovým zásobníkem; kapacita 64 kB představuje rezervu více než 300 ms i pro nejvyšší vzorkovací frekvenci. Analogové vstupy doplňují dva analogové výstupy s 12bitovými D/A převodníky, dva čítače, osm digitálních vstupů a osm digitálních výstupů. Analogové vstupy a výstupy jsou dostupné na konektoru Cannon 25, čítače pak na konektoru Cannon 9; oba jsou umístěny přímo na PC štítku karty.
Digitální porty využívají rozhraní s konektorem Cannon 9 (konektor s 8 signály doplněnými o GND).

Karta PCA-7228AS umožňuje vyvolat přerušení volitelně po ukončení každé sekvence vstupů (tzn. sestava vstupů definovaná ve scanovací logice) nebo při zaplnění paměti v úrovni 256B, 512B, 2kB, 8kB a 32kB.

Obrázek 25. Multifunkční měřicí karta PCA - 7228AS

Karta umožňuje definovat až 32 kanálů jako libovolnou kombinaci vstupů a napěťových rozsahů (vstupy se mohou i opakovat) včetně definování časového rozestupu. Pro případ dvou vstupů vzorkovaných frekvencí 1kHz, jsou měření vzdálena pouze o 0,01 ms. Karta umožňuje rovněž synchronní záznam čtyř digitálních signálů [TEDIA 2004].

Technické parametry karty viz Příloha H.

7.2. Multifunkční měřící PC karta PCA 7428 AS

Tato karta byla použita pro řízení modelu levitace feromagnetického předmětu v elektromagnetickém poli v následující kapitole.
Jediný rozdíl mezi kartou PCA-7428AS a kartou PCA-7228AS je ve velikosti rozlišení vstupních registrů, v případě karty PCA-7428AS je rozlišení vstupních registrů 14bitů na rozdíl od 12 bitů u karty PCA-7228AS.

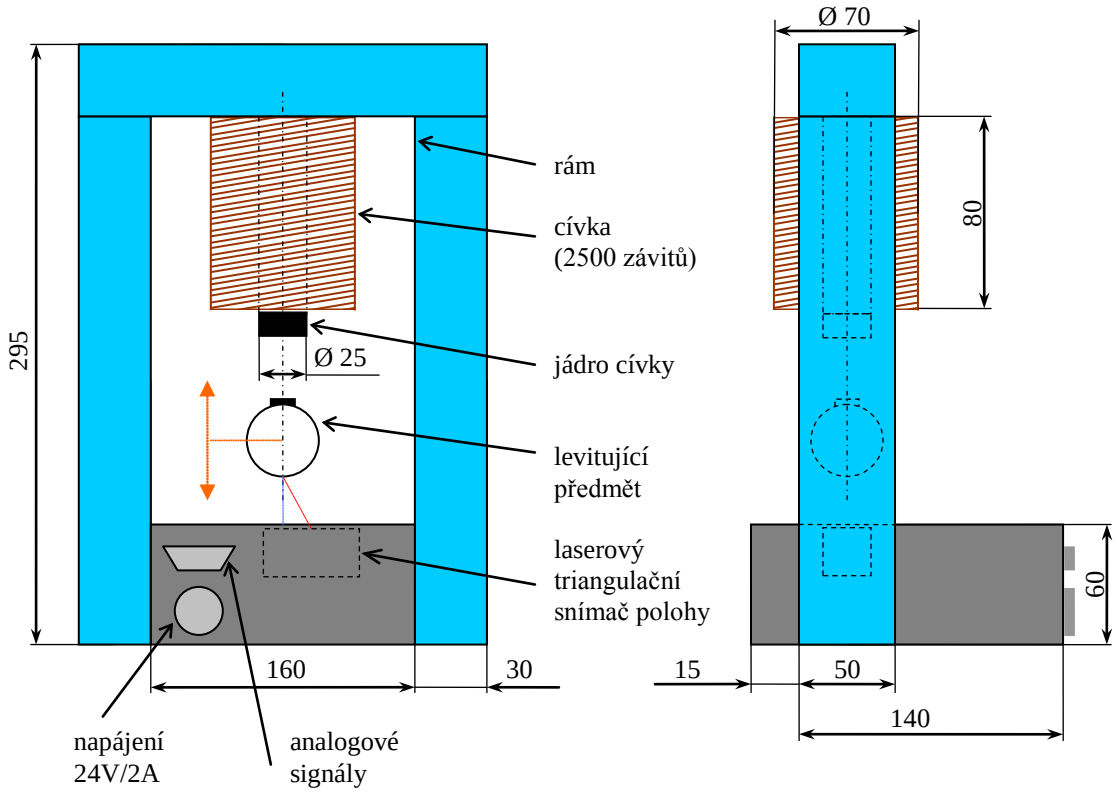
Technické parametry karty viz Příloha H.

8. Řízení modelu levitace pomocí doplňku RTX

Doplněk RTX operačního systému Windows, bylo nutné ověřit na reálném zařízení, model levitace feromagnetického předmětu v elektromagnetickém poli, byl již vytvořen v rámci projektu GAČR Dr. Kačmářem. Byl také využit v disertační práci Ing. Fojtíka [FOJTÍK 2004] a bakalářskou práci Ing. Chlebka [CHLEBEK 2005]. Tento model je charakteristický nelinearitou a velice malými časovými konstantami a potřebě provádět algoritmus řízení s krátkou vzorkovací periodou max. 1ms. Tento model je proto využít pro praktické ověření schopností systémů reálného času.

8.1. Technický popis modelu levitace

Model pracuje na principu vertikálního přitahování předmětu elektromagnetickým polem. Síla, jež působící na levitující předmět je vytvořena cívkou elektromagnetu. Změnou velikosti napětí na cívce elektromagnetu, se mění poloha levitujícího předmětu. Cívka je napájena speciální galvanickou elektronikou, ta mění akční veličiny z 0 ÷ 10 V na 0 ÷ 20 V (0 ÷ 2 A).



Obrázek 26: Technické schéma model levitace feromagnetického předmětu v elektromagnetickém poli [FOJTÍK 2004]

Tabulka 5: Rozměry cívky modelu levitace [FOJTÍK 2004]

délka	80 mm
vnější průměr	70 mm
počet závitů	2500
průměr drátů	2,1 mm
průměr jádra	25 mm
materiál jádra	ocel třídy 11

Obrázek 27. Model levitace feromagnetického předmětu v elektromagnetickém poli [FOJTÍK 2004]

Jako levitující předmět je použit pingpongový míček, na jehož horní straně je připevněn (přilepen) permanentní magnet. Velikost míčku zaručuje lepší snímání polohy levitujícího předmětu.

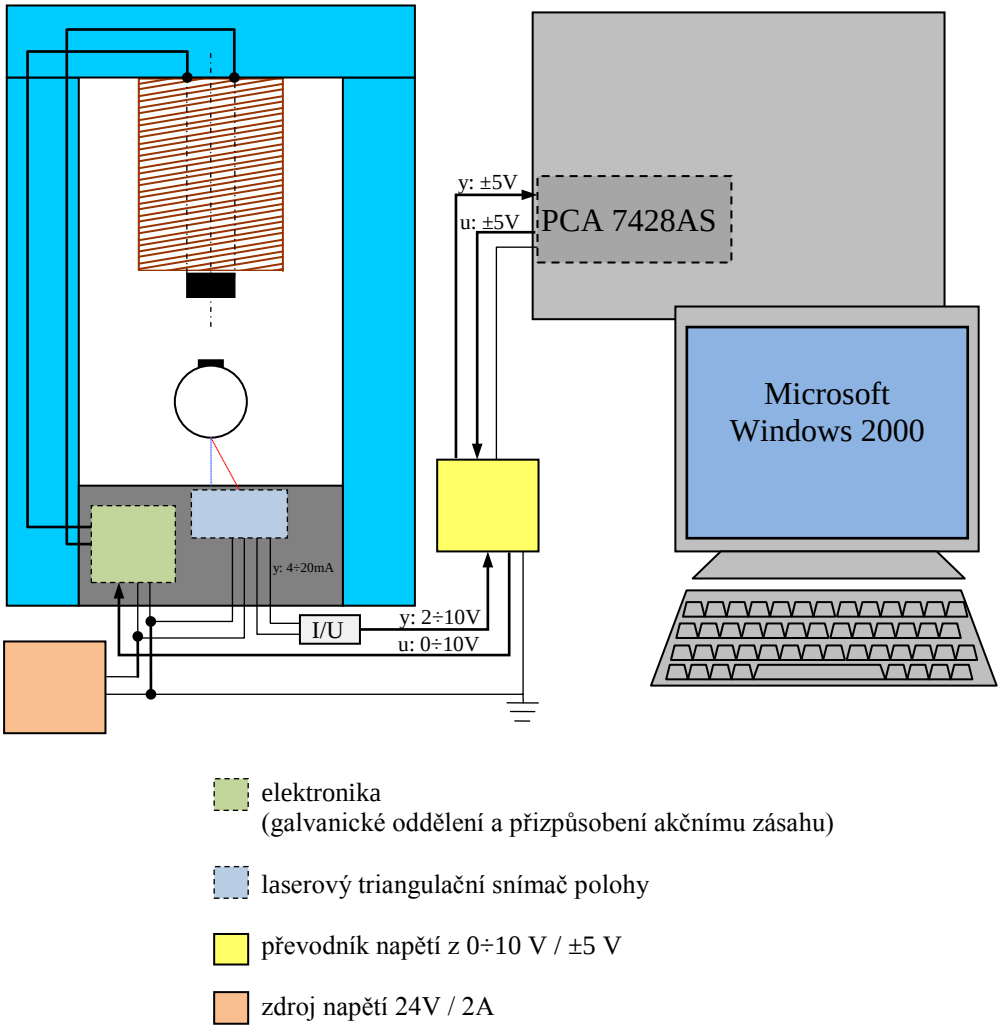
Pro měření polohy je použit laserový snímač polohy od firmy SICK.
Tento snímač pracuje na triangulačním principu. Výstupní signál polohy je realizován proudem v rozsahu 4 ÷ 20 mA pro vzdálenost 20 mm [FOJTÍK 2004].

Model levitace je propojen s PC přes multifunkční měřící kartu PCA-7428AS. Model pracuje s bipolárními hodnotami signálu 2 ÷ 10 V. Akční veličina musí do modelu vstupovat v rozsahu 0 ÷ 10 V.

Aby bylo využito plného rozsahu převodníku v multifunkční kartě, bylo nutné signál upravit. Za tímto účelem funguje samostatný převodník napětí, jež převádí analogové signály z rozsahu 0 ÷ 10 V na ± 5 V a opačně [FOJTÍK 2004].

Tabulka 6: Hardwarové a softwarové vybavení počítače řídicího model levitace

procesor (CPU)	Intel Pentium II 333 MHz
takt sběrnice	100 MHz
pevný disk	15 GB
připojení disku	Ultra DMA, IDE ATA 33(IRQ14)
operační systém	MS Windows 2000 + SP4
verze doplňku reálného času IntervalZero (Ardence)	6.5.1 (ověřena také 8.1.1)



Obrázek 28: Schéma modelu levitace a řídicího PC [FOJTÍK 2004]

8.2. Matematický model levitace

Uvedený matematický model, odpovídá modelu levitace za předpokladu, že levitující předmět se pohybuje pouze ve vertikálním směru, permeabilita jádra je nekonečná a magnetickou hysterezi zanedbáme [FOJTÍK 2004].

$$m\ddot{x} = mg + \frac{1}{2} i^2 \frac{\partial L(x)}{\partial x}, \tag{8.1}$$

$$u = Ri + \frac{d}{dt}\left[L(x) \cdot i\right], \tag{8.2}$$

$$L(x) = \frac{Q}{X_\infty + x} + L_\infty, \tag{8.3}$$

kde je - m – hmotnost levitujícího objektu [kg], x – vzdálenost mezi levitujícím objektem a magnetem [m], u – elektrické napětí [V], i – elektrický proud [A], R – elektrický odpor vinutí [Ω], $L(x)$ – induktance vinutí [H], Q, L_∞, X_∞ parametry dané fyzikálními charakteristikami vinutí, jádra a levitujícího předmětu [H·m, H, m].

Úpravou rovnic (8.1), (8.2), (8.3) a derivacemi podle času a polohy, dostaneme rovnice (8.1) a (8.2) do tvaru:

$$m\ddot{x} = mg - \frac{Q}{2} \frac{1}{\left(X_\infty + x\right)} i^2, \tag{8.4}$$

$$u = Ri - \frac{Q}{\left(X_\infty + x\right)} \frac{d}{dt} i + \left(\frac{Q}{\left(X_\infty + x\right)} + L_\infty \right) \frac{d}{dt}, \tag{8.5}$$

Následnou úpravou těchto rovnic a zavedením stavových proměnných $x_1 = x, \quad x_2 = \dot{x}, \quad x_3 = \ddot{x}$, dostaneme stavové vyjádření:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= f_3(\mathbf{x}) + g_3(\mathbf{x}) u \end{aligned} \tag{8.6}$$

kde je – u – elektrické napětí [V], $f_3(\mathbf{x})$ a $g_3(\mathbf{x})$ - obecné nelineární funkce

stavových proměnných, které jsou vyjádřeny následujícím vztahy:

$$f_3(\mathbf{x}) = \left[\frac{2R}{\frac{Q}{(X_\infty + x_1)} + L_\infty} + \frac{2x_2}{(X_\infty + x_1)} - \frac{2Qx_2}{(X_\infty + x_1)^2 \left(\frac{Q}{(X_\infty + x_1)} + L_\infty \right)} \right] (q - x_3), \tag{8.7}$$

$$g_3(\mathbf{x}) = -\sqrt{\frac{2Q}{m}} \frac{\sqrt{g - x_3}}{(X_\infty + x_1)\left(\frac{Q}{(X_\infty + x_1)} \right) + L_\infty}, \tag{8.8}$$

Z rovnic je zřejmé, že systém je vysoce nelineární [FOJTÍK 2004].

8.3. Regulátor řízení

Základem algoritmu řízení je PID (PSD) polohový regulátor, který je jen číslicovou verzí PID regulátoru.

8.3.1. PID (PSD) regulátor s filtrací derivační složky

Tento typ PID regulátoru byl naprogramován jako řídicí regulátor levitace. Regulátor je charakteristický malým zpožděním. To zaručuje dobrou regulaci prudkých změn žádané veličiny. Dále odstraňuje šum vysokofrekvenčních složek, aby nedocházelo k velkým změnám, je nahrazena spojitá derivace derivačním členem se zpožděním prvního řádu (8.9):

$$T_D \frac{s}{T_f s + 1} \tag{8.9}$$

kde je – T_D – derivační konstanta, T_f – časová konstanta filtru [s].

Regulátor PID lze následně vyjádřit takto:

$$u(t) = k_P \left[e(t) + \frac{1}{T_I} \int_0^t e(\tau) d\tau + T_D \frac{de_f(t)}{dt} \right], \tag{8.10}$$

kde je – k_p – zesílení regulátoru, *e(t)* – regulační odchylka, *u(t)* – akční veličina, T_I – integrační časová konstanta [s], *t* - čas [s].

Při nahrazení spojitého integračního členu obdélníky z leva se rovnice převede do tvaru:

$$u(kT) = k_P \left\{ e(kT) + \frac{T}{T_I} \sum_{i=1}^{k} e(iT) + \frac{T_D}{T} \left\{ e_f(kT) - e_f\left[(k-1)T\right] \right\} \right\} \tag{8.11}$$

kde je – T – vzorkovací perioda číslicového regulátoru; k – relativní diskrétní čas; $e_f(t)$ – filtrovaná regulační odchylka $e(t)$.

Filtrovaná regulační odchylka $e_f(kT)$ se vypočte vztahem:

$$e_f(kT) = (1-a) \cdot e_f\left[(k-1)T\right] + a \cdot e(kT) \tag{8.12}$$

kde je – a – koeficient filtrace [FOJTÍK 2004].

Optimální parametry regulátoru byly získány z disertační práce Ing. Davida Fojtíka, Ph.D [FOJTÍK 2004].

Tvar regulátoru ve zdrojovém kódu je následující:

```
void RTFCNDCL PID(PVOID addr)
{// pomocna promenna – udaje o merici karte
      PPCI_COMMON_CONFIG  pciData = (PPCI_COMMON_CONFIG) &buffer;
      ULONG       low = 0; // definice a nastaveni pomocnych promennych
      ULONG       high = 0;
      LONG        ven = 0;
      LONG        maxW = 0xFFF;
      LONG        Ul = 0;
      double      P = kp;           // definice pomocnych promennych
PID
      double      I = kp * (T/Ti);
      double      D = kp * (Td/T);
// nastaveni registru CWReg na hodnotu SCWReg (0x40) – karta pracuje
se softwerovym spoustenim a IRQ logika je vypnuta
      RtWritePortUlong(CWReg, SCWReg);
// cekani az bude registr StatusReg nulovy (karta je inicializovana
a pripravena k mereni)
      while((RtReadPortUlong(StatusReg)&0xFF) != 0x0);
// zapisem jakych koliv dat do registru SWTrigRed dojde ke spusteni
merici sekvence při softwerovem spousteni
      RtWritePortUlong(SWTrigReg, SSWTrigReg);
// cekani az bude reg. StatusReg = 0 (karta dokoncila merici sekvenci)
      while((RtReadPortUlong(StatusReg)&0xFF) != 0x0);
// precteni dolni casti namerene hodnoty z registru merici karty
      low = (RtReadPortUlong(REG_LO) & 0xFF);
// precteni horni casti namerene hodnoty z registru merici karty
      high = (RtReadPortUlong(REG_HI) & 0xFF);
// ulozeni horni namerene casti hodnoty a spojeni s dolni casti
      Y = ((high<<8)|low);
// zapisem hodnoty 0x0 do CWReg dojde k ukonceni merici sekvence
      RtWritePortUlong(CWReg, XCWReg);
```

```
// prepocet Y na volty a ulozeni do sdilene pameti
      Y = ((10*Y/0xFFFC)-5);
// vypocet E z hodnoty ulozene ve sdilene pameti a namerene hodnoty Y
      E = W - Y;
// vypocet hodnoty Ef z hodnot jez jsou ulozene ve sdilene pameti
      Ef = (1-a)*Efmin + a*E;
// vypocet SumE z hodnot ve sdilene pameti
      SumE += E;
// vypocet U z hodnot ve sdilene pameti
      U = P*E + I*SumE + D*(Ef - Efmin);
// vypocet Efmin z hodnot ve sdilene pameti
      Efmin = Ef;
// prepocet U z voltu na hodnotu kterou bude mozno zapsat na vystup
      U = (U*0xFFF/10)+(0xFFF/2);
// vypocet zaokrouhleni (zvetseni presnosti)
      U = (int)(U + 0.5);
      Ul = (LONG)U;
// overeni zda hodnota U je v rozdezi do maximalni hodnoty, kterou lze
zapsat na vystup merici karty
      if(Ul > maxW){
// pokud je hodnota Ul vetsi nez maximalni mozna hodnota, kterou lze
zapsat na vystup merici karty, tak je na vystup zapsana maximalni
hodnota zapisu, která je rozdelena na dve casti a zapsana na vystup
v poradi nizzsi/vyssi
            RtWritePortUlong(DACReg0l, 0xff);
            RtWritePortUlong(DACReg0h, 0xf);
      }else{
// pokud je hodnota pro zapis na vystup mensi nez maximalni mozna, tak
se rozdeli na dve casti a zapise postupne do registru pro vystup
v poradi nizssi/vyssi
            ven = (Ul & 0xff);
            RtWritePortUlong(DACReg0l, ven);
            RtWritePortUlong(DACReg0h, Ul>>8);
      }
}
```

8.4. Aplikace řídicí model levitace

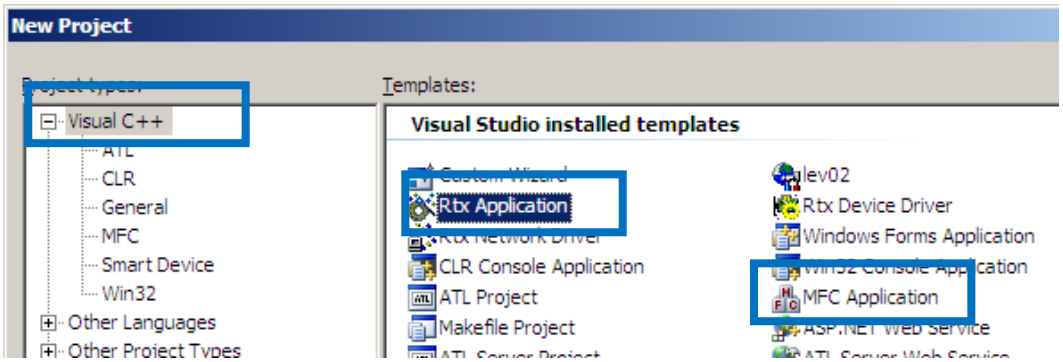
Aplikace byla vyvíjena ve vývojovém prostředí Microsoft Visual Studio 2005. I když již byla v době vývoje k dispozici verze Visual Studia 2008, musela být použita verze 2005, jelikož doplněk RTX ve verzi 8.1.1. podporuje pouze Visual Studio do verze 2005. Jako měřicí karty byly zkoušeny a použity karty PCA 7228AS a PCA 7428AS.

Základem bylo vytvoření programu, jež bude regulovat model levitace v prostředí RTX a bude ovládána z prostředí operačního systému Microsoft Windows.

Za tímto účelem byly vytvořeny dva programy:

- program uživatelského rozhranní v prostředí MS Windows,
- program řídicí model levitace v prostředí RTX.

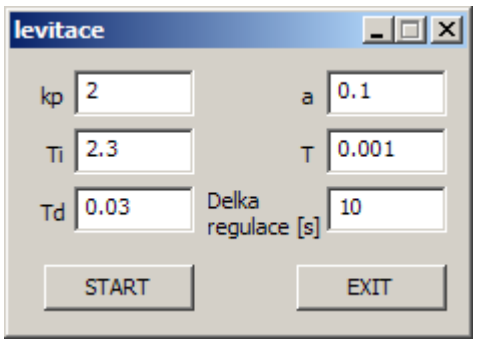
Program uživatelského rozhraní v MS Windows spouští a ukončuje program v prostředí RTX. Program uživatelského rozhraní byl vytvořen v jazyce C++ využívajíc Microsoft Foundation Class Library (MFC) a program řídicí model levitace v jazyce C++ využívajíc RTSS prostředí instalované ve Visual Studiu.



Obrázek 29. Zvolení typu programů pro aplikaci levitace v prostředí Visual Studia 2005

8.4.1. Program uživatelského rozhraní

Program je tvořen dvěma tlačítky (START a EXIT) a šesti textboxy, pro zadávání hodnot pro regulaci (kp, Ti, Td, a, T a délky regulace v sekundách). Při spuštění programu uživatelského rozhraní se do textboxů automaticky doplní optimální hodnoty pro levitaci, které byly vypočteny a ověřeny dlouhodobým testováním (kp = 2; Ti = 3.2; Td = 0.03; a = 0.1; T = 0.001). Tlačítko START spouští klientskou aplikaci. Při spuštění klientské aplikace hlavní aplikace vytvoří příkazový řádek, do kterého vloží hodnoty parametrů z textboxů. A klientské aplikaci je předá jako spouštěcí argumenty.



Obrázek 30: Dialogové okno uživatelského rozhraní levitace v prostředí MS Windows

8.4.2. Program vlastního řízení v prostředí RTX

Program řízení při startu nejprve ověří, zda dostal v příkazové řádce potřebné argumenty (ty si uloží pro nastavení regulace), prohledá PCI sloty, zdali je v počítači měřicí karta PCA – 7428AS, provede inicializaci karty, vytvoří časovače a začne provádět periodickou regulaci modelu levitace.

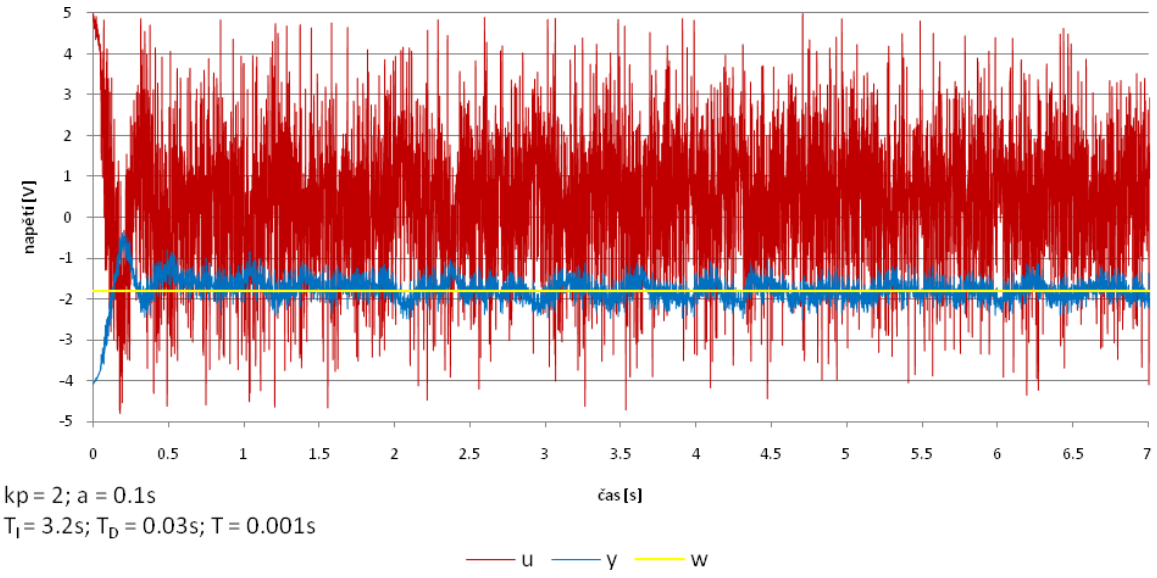
Změna parametrů PID regulátoru se provádí v programu uživatelského rozhraní, po dokončení předchozího cyklu regulace. Po ukončení spuštěného cyklu regulace se zpřístupní textboxy a je umožněno přednastavit hodnoty regulace nebo případně ukončit program.

Opětovné spuštění regulace modelu levitace se provede stisknutím tlačítka START. Tlačítko EXIT ukončí program uživatelského rozhraní.

Záznam průběhu regulace je ukládán do txt souboru, který je generován programem řízení na hard disk.
Popis registrů měřící karty a zdrojové soubory programu levitace viz Příloha F.

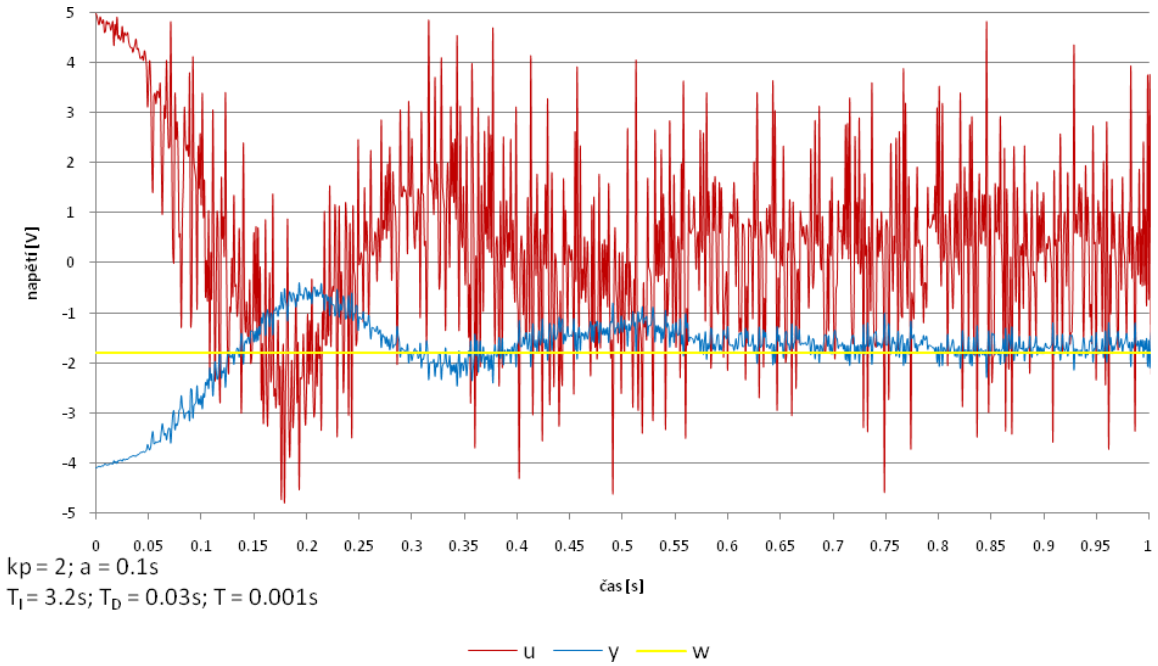
8.4.3. Grafy průběhu regulace modelu levitace

V průběhu testování regulace byly ověřeny různé hodnoty nastavení PID regulátoru. Jako nejuspokojivější pro kvalitu regulace byly zvoleny hodnoty kp = 2, a = 0.1s, T_I = 3.2s, T_D = 0.03s, T = 0.01s, jejichž grafy jsou zobrazeny níže.



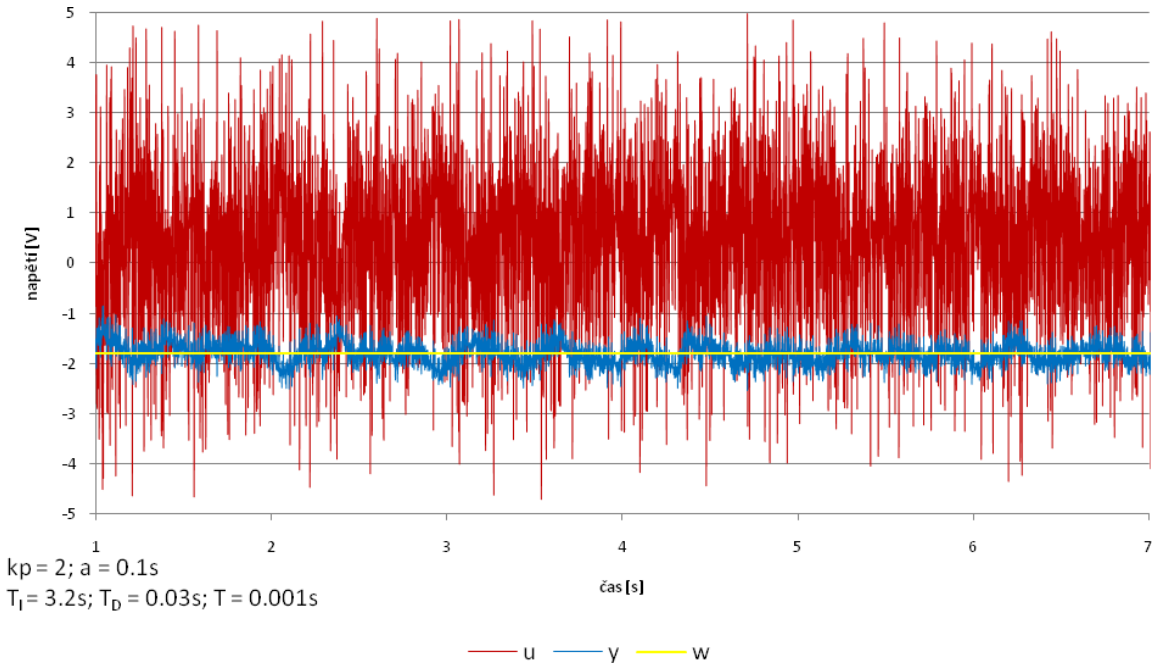
Obrázek 31. Průběh regulace PID polohovým regulátorem s filtrem D složky

Obrázek 32 zobrazuje průběh začátku regulace, kdy dochází k pozvednutí předmětu z pevné základny do výšky odpovídající hodnotě - 1.8 V.



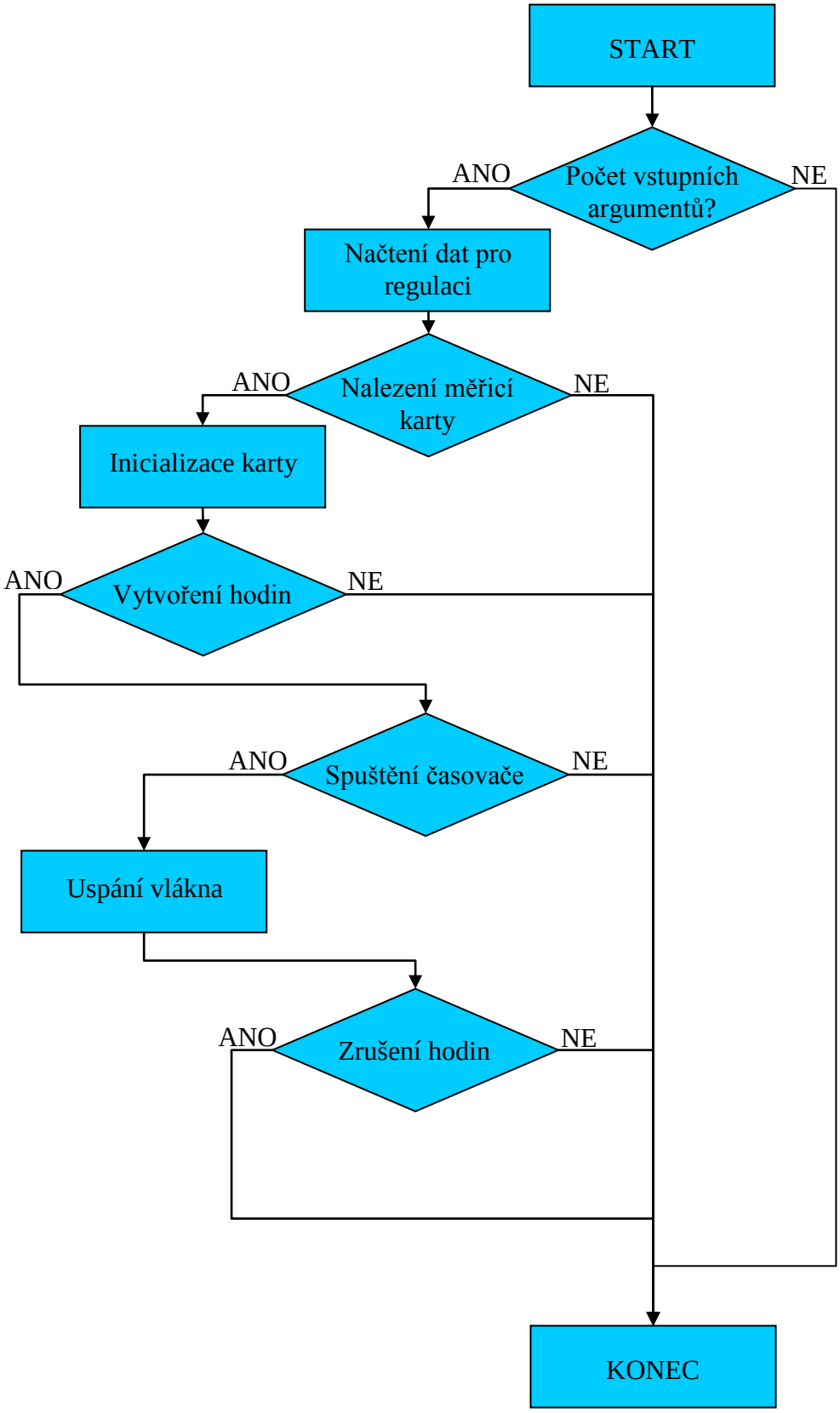
Obrázek 32. Průběh regulace PID polohovým regulátorem s filtrem D složky, začátek regulace (čas 0 až 1 sekunda)

Obrázek 33 zobrazuje průběh regulace v ustáleném stavu.



Obrázek 33. Průběh regulace PID polohovým regulátorem s filtrem D složky, ustálený stav

Vývojový diagram klientského programu levitace:



Vývojový diagram 3: Klientská aplikace řídicí model levitace

9. Zhodnocení

Diplomová práce se zabývá základními vlastnostmi operačních systémů, jejich strukturou a použitím v současných zařízeních, která nás obklopují.
Byly popsány vybrané systémy a operační systémy reálného času (RTOS) a srovnány jejich vlastnosti s operačními systémy Windows XP/Vista.
V práci byly definovány požadavky, které musí operační systém reálného času splňovat, aby se mohl považovat za systém reálného času.
Dále bylo provedeno rozdělení operačních systémů reálného času na hard a soft systémy reálného času, se stručným přehledem nejznámějších operačních systémů reálného času, se stručným popisem jejich vlastnosti, struktur a použití.
Detailně byl popsán produkt RTX od společnosti IntervalZero, ve verzích 6.5.1 a 8.1.1, který dává operačnímu systému MS Windows schopnosti operačního systému reálného času, pro řízení aplikací v reálném čase.
Pro práci byla vytvořena ukázková aplikace, která přenáší data z jednoho počítače na druhý počítač přes paralelní port, v daných časových intervalech. Tato aplikace byla vytvořena jak pro MS Windows, tak pro MS Windows s doplňkem RTX. Následně se provedlo srovnání výsledků mezi aplikacemi, jež přenášela data přes OS Windows a RTOS (bez a s doplňkem RTX). Porovnáním výsledků bylo dokázáno, že aplikace běžící pod RTX dokáže provádět svoji činnost se stoprocentní správností, na rozdíl od systému MS Windows.
Práce popisuje způsoby práce s doplňkem RTX a to, jakým způsobem se přidávají a odebírají zařízení pod a ze správy RTX, což je důležité pro správný běh programů pod subsystémem RTX. Diplomová práce uvádí, jakým způsobem se vytváří aplikace pro RTX (subsystém reálného času).
Jako další ukázka schopností doplňku RTX, byla vytvořena aplikace řídící model levitace feromagnetického předmětu v elektromagnetickém poli.
Tato aplikace je tvořena dvěma programy. Jeden je uživatelské rozhraní, pracující v prostředí MS Windows a druhý je program vlastní levitace, který pracuje v prostředí RTX. Program pracující v prostředí MS Windows předá přes parametry příkazové řádky argumenty programu řídicím model levitace a ten

následně řídí model levitace. Naměřená data se ukládají na pevný disk do textového souboru, z něj se s nimi dá standardně pracovat.
Do budoucna by se vytvořená aplikace mohla přeprogramovat tak, aby využívala MEM (paměťový) přístupu k multifunkční kartě, jelikož vytvořená aplikace řídicí model levitace, využívá I/O přístup. Do aplikace levitace by se mohla doprogramovat část, která by uživateli nabídla možnost, vykreslovat aktuální stav levitace s možností okamžité práce s naměřenými daty. Dalším případným vylepšením by mohlo být využívání sdílené paměti mezi oběma programy.

Seznam odborné literatury

ARDENCE 2002 [A]. *RTX Documentation* [online]. Ardence, Inc. PDF formát. Dostupné z: <URL: http://www.ardence.com/embedded/products.aspx?ID=214>.

ARDENCE 2002 [B]. *PharLap Documentation* [online]. Ardence, Inc. PDF formát. Dostupné z: <URL: http://www.ardence.com/Embedded/products.aspx?ID=71>.

AUTOMA č. 7. 2004. Dostupné z: <URL: http://www.odbornecasopisy.cz/automa/2004/au070432.htm>.

CHLEBEK, M. 2005. *Ovládací aplikace modelu levitace předmětu v elektromagnetickém poli.* Bakalářská práce. Ostrava: VŠB-TU Ostrava

COMP.REALTIME 2004A. Diskusní fórum, dostupné z: <URL: http://www.realtime-info.be/encyc/ techno/publi/faq/rtfaq.htm#Windows_NT>

COMP.REALTIME 2004B. Webový informační server dostupný z: <URL: http://www.realtimeinfo.be/encyc/ techno/publi/faq/rtfaq.htm#WindowsNT>.

CRAIG PEACOCK'S 1998 [A]. *Interfacing the Standard Parallel Port* [online]. Craig Peacock February 1998. PDF formát. Dostupné z <URL: http://www.beyondlogic.org>.

CYBERSPACE 2009. Webový informační server dostupný z: <URL: http://www.cyberspace.cz>

DATAPARTNER 2008. Webový informační server, dostupný z: <URL: http://www.datapartner.cz>.

FARANA, R., SMUTNÝ, L., VÍTEČEK, A. 2004. *Zpracování odborných textů z oblasti automatizace a informatiky*. 2. vyd. Ostrava: VŠB-TU Ostrava, 2004. 68 s. ISBN 80-7078-737-6.

FOJTÍK 2004. *Tvorba systémových modulů pro podporu řízení v operačním systému MS Windows NT*. Disertační práce. Ostrava: VŠB-TU Ostrava.

HUMUSOFT 1997. *AD 512 User's manual*. Firemní literatura Humusoft s.r.o.

MSDN 2009. Webový informační server dostupný z: <URL: http://msdn.microsoft.com>.

REISNER, L. 2002. *Windows XP a RTX – platforma reálného času*. Automatizace, 2002, roč. 45, č. 8, str. 464-465.

ŠVARC, I. 1993. *Teorie automatického řízení II.* Brno: Vysoké učení technické v Brně, 1993, třetí přepracované vydání. ISBN 80-214-0550-3.

TEDIA 2004 [A]. *Uživatelská příručka PCA-7228*. Firemní literatura Tedia s.r.o. 2004. Též dostupné z: <URL: http://www.tedia.cz/cz/manualy/pca7000a.pdf>

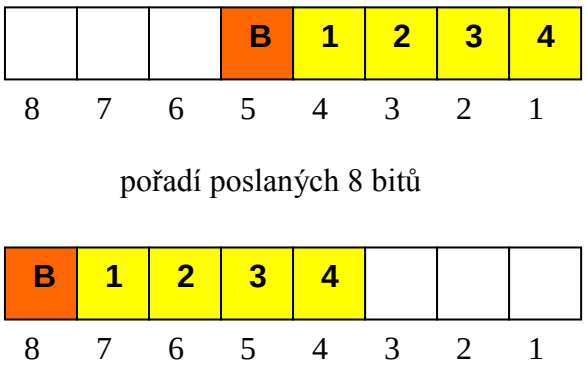
TEDIA 2009 [B]. Dostupné z <URL: http://www.tedia.cz/produkty/pca7228a.html>.

WIKIPEDIA 2008. Dostupné z <URL: http://www.wikipedia.org>.

Příloha A. Úpravy při čtení a zápise na standardní paralelní port

Komunikace mezi počítači, nebo počítačem a externím zařízením může probíhat prostřednictvím různých rozhraní. Vedle síťového spojení, USB, sériové linky je to i LPT.

Vzhledem k tomu že na paralelní port můžeme sice vyslat celý byte naráz (všech osm bitů současně), ale už nemůžeme všech osm bitů z LPT přečíst současně (platí pro standardní paralelní port). Pro čtení máme k dispozici pouze pět bitů, z toho jeden negující tzv. busi bit. To znamená, že pokud na něj vyšleme 0, přečteme z něj na druhé straně 1. Abychom tedy mohli přenést celý byte, musíme jej rozdělit na dvě půlky po čtyřech bitech. Každou půlku pošleme zvlášť a potom sečteme dohromady, přičemž busi bit používáme k tomu, abychom byly schopni rozpoznat, která půlka byte je zrovna vysílána druhým PC. Když tedy při vysílání první poloviny nastavíme na busi bit např. 1, pak musíme pro druhou polovinu nastavit busi bit na 0. Dále je třeba si uvědomit, že vyslaný byte a přijatý mají navzájem posunuté bity o 3 směrem vlevo.

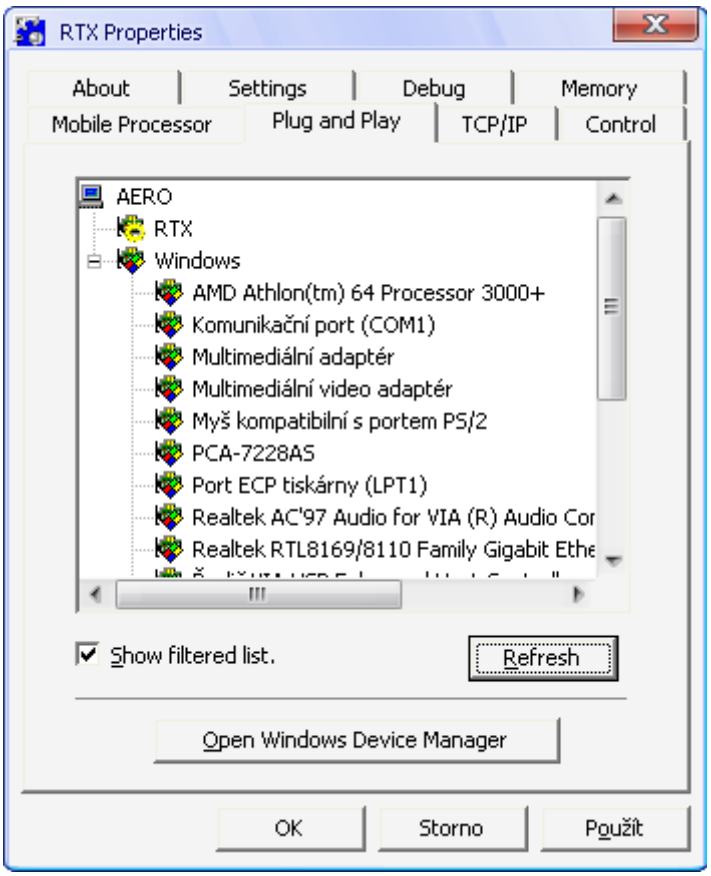


Obrázek 34. Schéma přenášených bitů

Příloha B. Přidání zařízení pod správu RTX

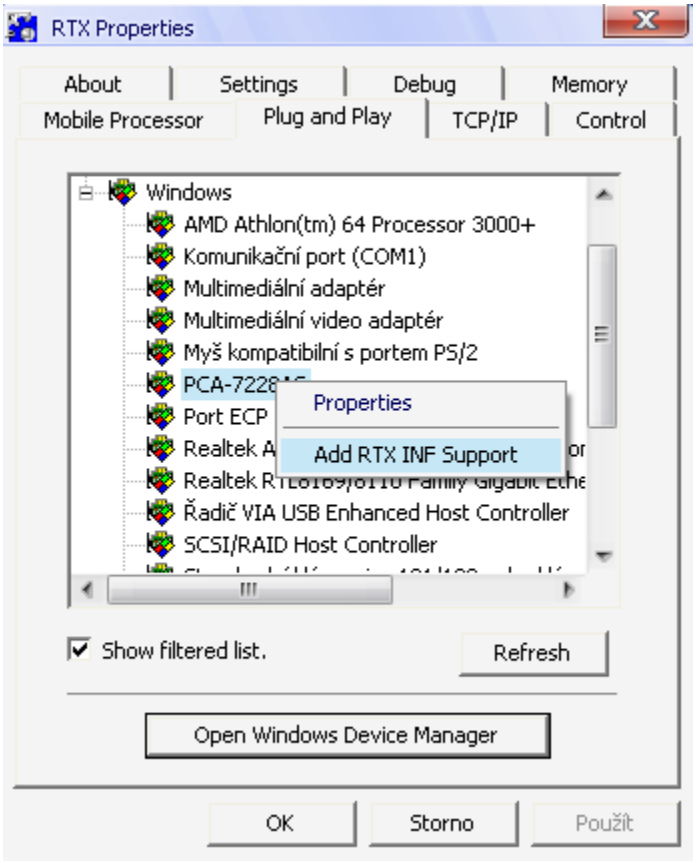
Přidání zařízení je nutné pro správnou funkci RTX programu, po přidání zařízení pod zprávu RTX je zařízení dostupné i z prostředí windows, ale při výskytu BSOD aplikace pod RTX funguje i nadále (postup lze provést až po nainstalování doplňku RTX do OS).

1. Přes Windows Start menu a Programs | Ardence RTX | RTX | RTX Properties se proklikáme k RTX Properties control panel.
2. V okně RTX Properties control, vyberte záložku Plug and Play.
   Tato záložka obsahuje seznam aktuálních Plug and Play (Pnp) zařízení v systému. Jestliže je zaškrtnuto políčko Show filtered list, tak se v seznamu zobrazí jen nejužívanější zařízení systému.
   K zobrazení všech zařízení odškrtnutí zrušte. K aktualizaci seznamu zařízení klikněte na tlačítko Refresh.



Obrázek 35: Okno RTX Properties na záložce Plug and Play

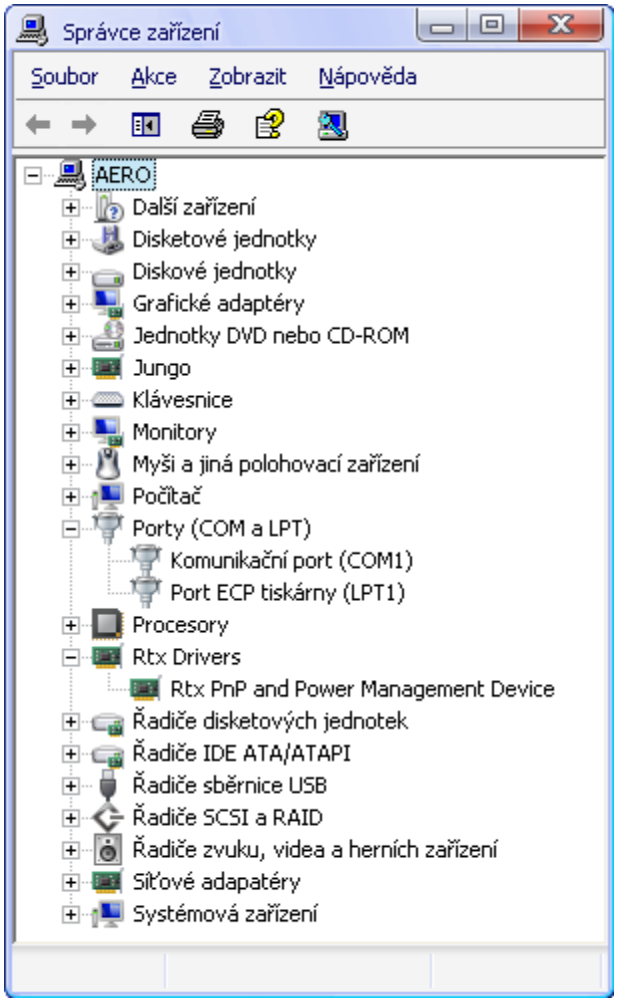
3. Pravým kliknutím na zařízení z Windows seznamu se vám zobrazí nabídka Vlastností zařízení (Properties) a Add RTX INF Support. Zvolte Add RTX INF Support.



Obrázek 36: Přidání RTX INF podpory vybranému zařízení

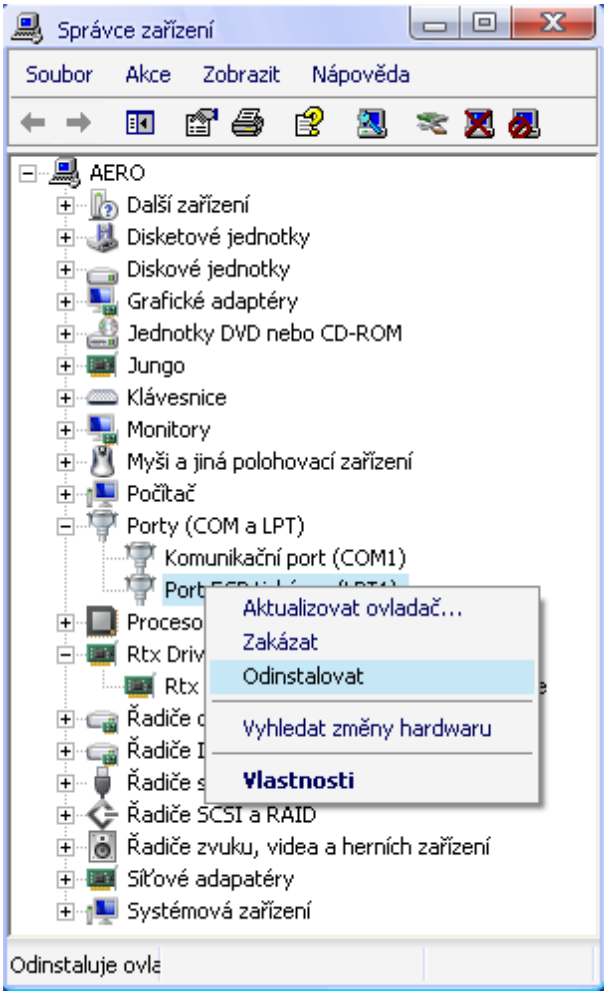
Toto přidání a přesunutí zařízení probíhá u LPT portu bez jakýchkoliv problémů i po nainstalování ovladačů (ovladače pro OS windows jsou pro LPT obvykle nainstalovány už během instalace OS).

4. Klikněte Apply.
5. Klikněte na Open Windows Device Manager k zobrazení Windows Device Manager.



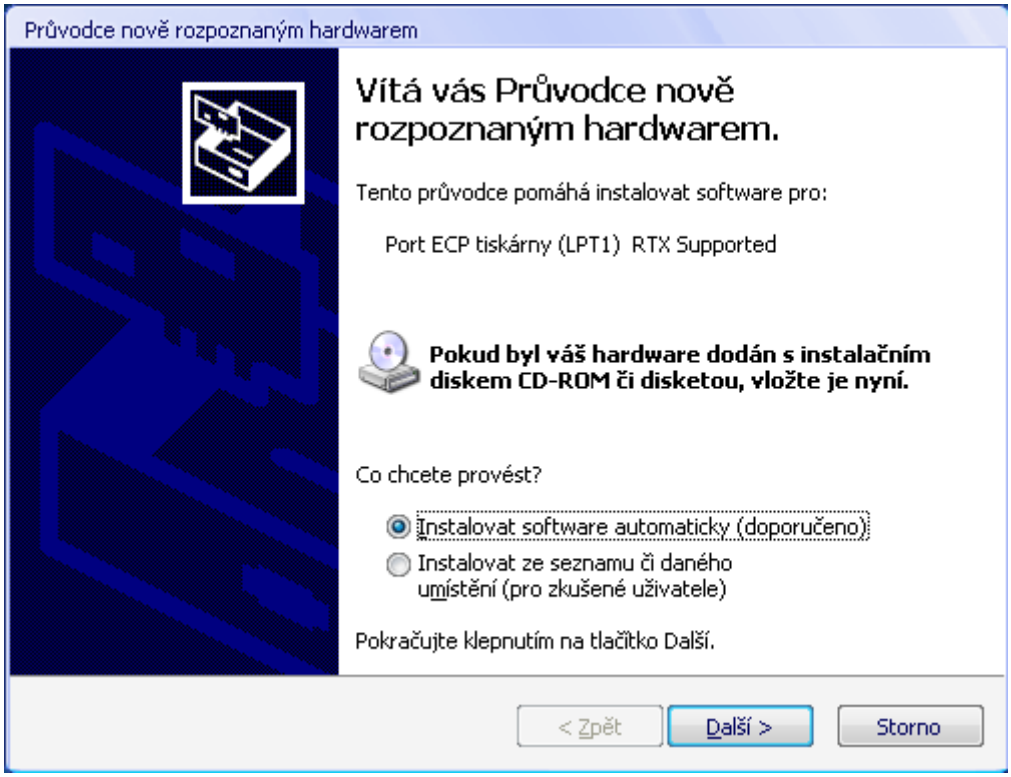
Obrázek 37: Okno Windows Device Manager

6. Najděte a klikněte pravým tlačítkem na zařízení, kterému jste právě přidali RTX INF support, a zvolte Uninstall.



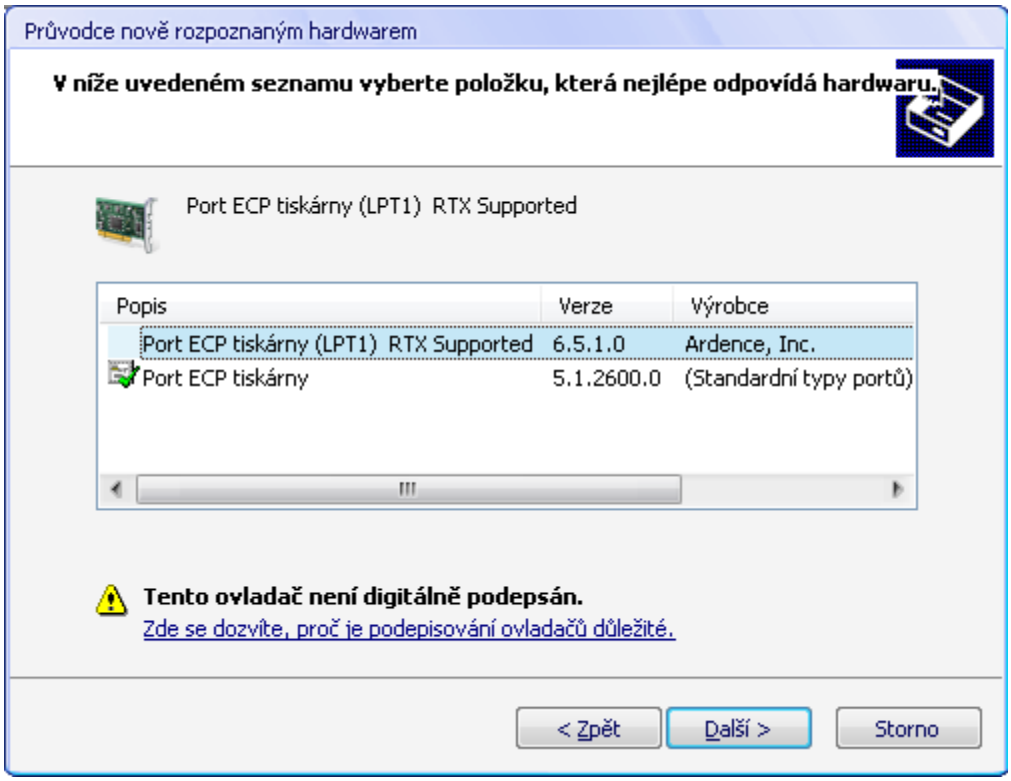
Obrázek 38: Odinstalace zařízení

7. Po zobrazení dialogového okna k odinstalaci zařízení ze systému. Klikněte na OK k odinstalaci zařízení.
8. V okně Windows Device Manager v menu klikněte na ikonu Scan for hardware changes. Poté se vám zobrazí průvodce přidáním nového zařízení.
9. Zvolte automatickou instalaci a klikněte na tlačítko Next. Průvodce vám nyní nabídne seznam možných ovladačů pro toto zařízení.



Obrázek 39: Instalace ovladače pro zařízení

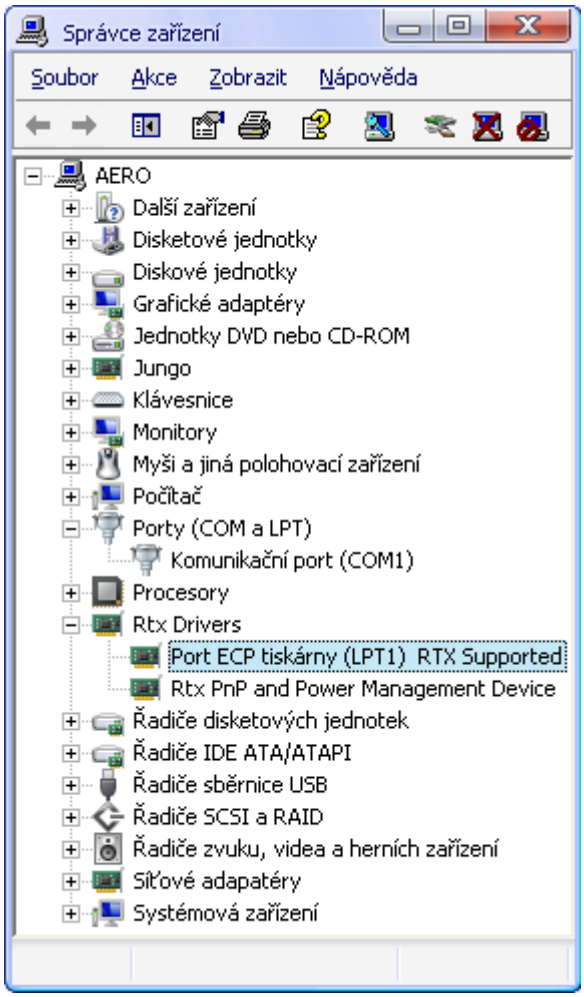
10. Zvolte RTX Supported a klikněte na tlačítko Next.



Obrázek 40: Zvolení ovladače RTX pro zařízení

11. Až dokončí průvodce instalaci zařízení, klikněte na tlačítko Finis.

    Po přesunutí do RTX Properties a kliknutí na tlačítko Refresh se již zařízení zobrazí v seznamu Rtx Drivers.

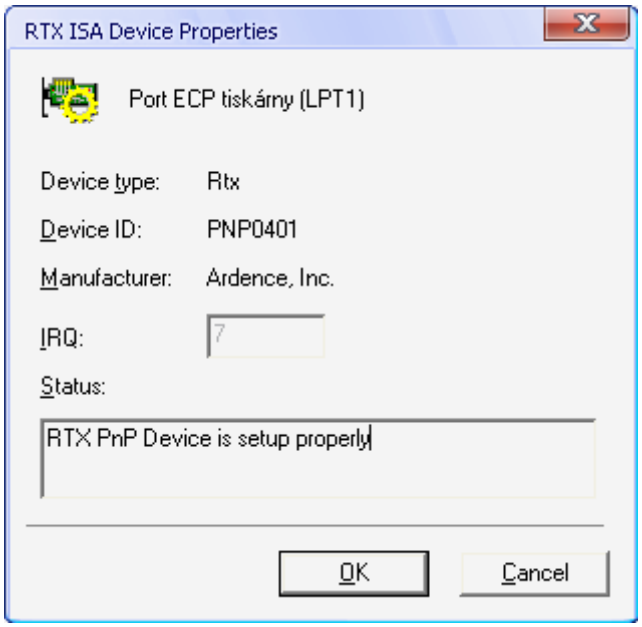


Obrázek 41: Zobrazení Windows Device Manager a přeinstalovaného zařízení

12. Nyní musíte restartovat počítač (v OS Windows XP Professional není restart nutný).

Příloha C. Zobrazení vlastností RTX zařízení

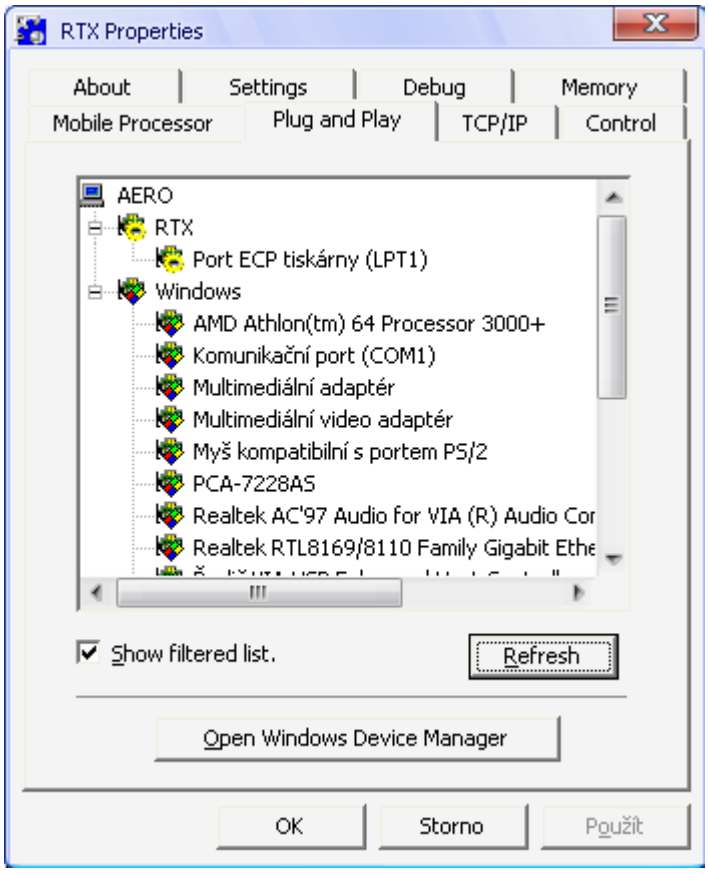
1. Přes Windows Start menu a Programs | Ardence RTX | RTX | RTX Properties se proklikáme k RTX Properties control panel.
2. V okně RTX Properties control, vyberte záložku Plug and Play. Tato záložka obsahuje seznam aktuálních Plug and Play (Pnp) zařízení v systému. Jestliže je zaškrtnuto políčko Show filtered list, tak se v seznamu zobrazí jen nejužívanější zařízení systému. K zobrazení všech zařízení odškrtnutí zrušte. K aktualizaci seznamu zařízení klikněte na tlačítko Refresh.
3. Dvojitým kliknutím na zařízení, jehož vlastnosti chcete vidět, se nám zobrazí okno s vlastnostmi zařízení.



Obrázek 42: Zobrazení vlastností RTX zařízení

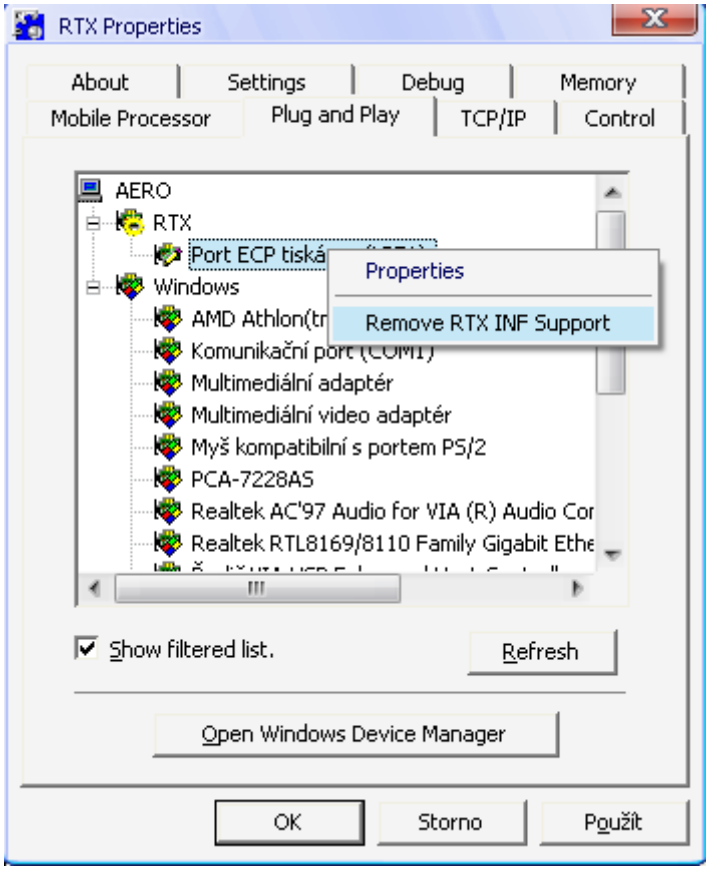
Příloha D. Odebrání zařízení RTX a vrácení pod správu OS Windows

1. Přes Windows Start menu a Programs | Ardence RTX | RTX | RTX Properties se proklikáme k RTX Properties control panel.
2. V okně RTX Properties control, vyberte záložku Plug and Play. Tato záložka obsahuje seznam aktuálních Plug and Play (Pnp) zařízení v systému. Jestliže je zaškrtnuto políčko Show filtered list, tak se v seznamu zobrazí jen nejužívanější zařízení systému. K zobrazení všech zařízení odškrtnutí zrušte. K aktualizaci seznamu zařízení klikněte na tlačítko *Refresh*.



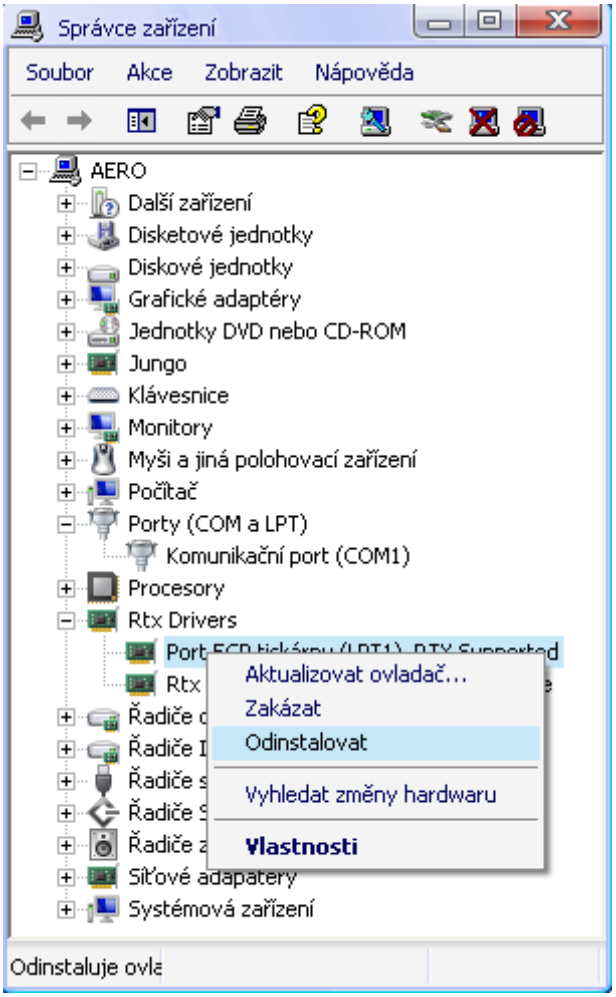
Obrázek 43: Okno RTX Properties s nainstalovaným zařízením pod RTX

3. Pravým kliknutím na RTX zařízení jež chceme odebrat, se nám zobrazí nabídka Remove RTX INF Support, tu zvolte pro odebrání zařízení z RTX.



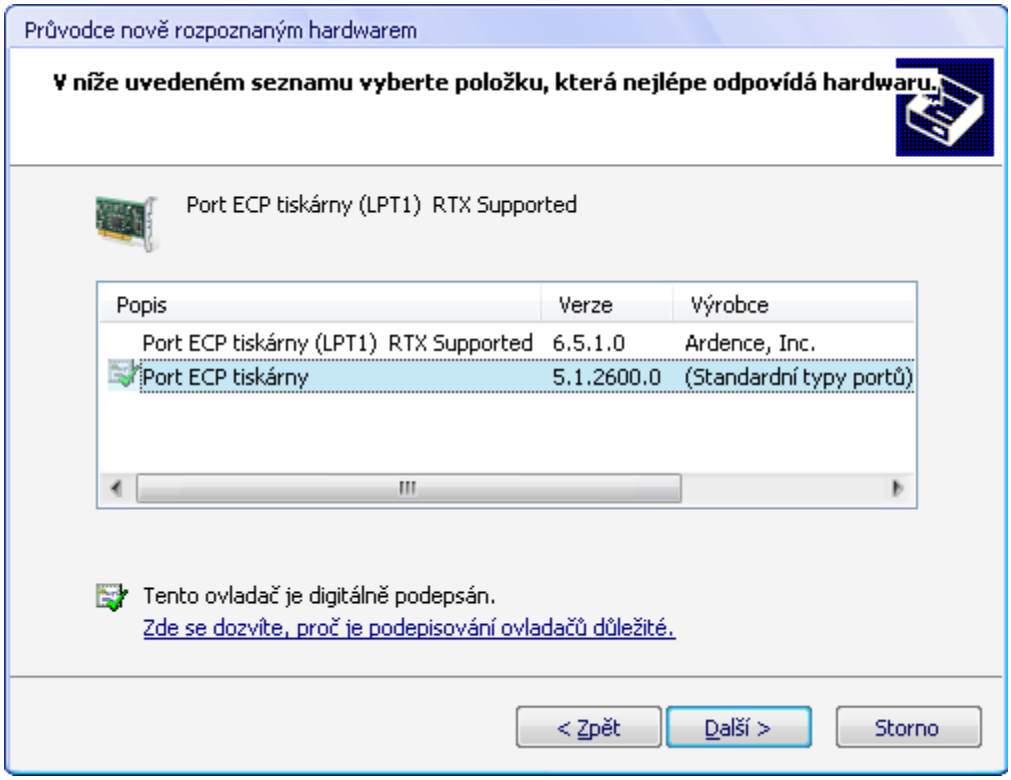
Obrázek 44: Odebrání RTX INF podpory RTX zařízení

4. Odklikněte Apply.
5. Klikněte na Open Windows Device Manager k zobrazení Windows Device Manager.
6. Nalezněte a pravým kliknutím označte zařízení, jež chcete odstranit z RTX a zvolte Uninstall.



Obrázek 45: Odinstalace RTX zařízení ve Windows Device Manager

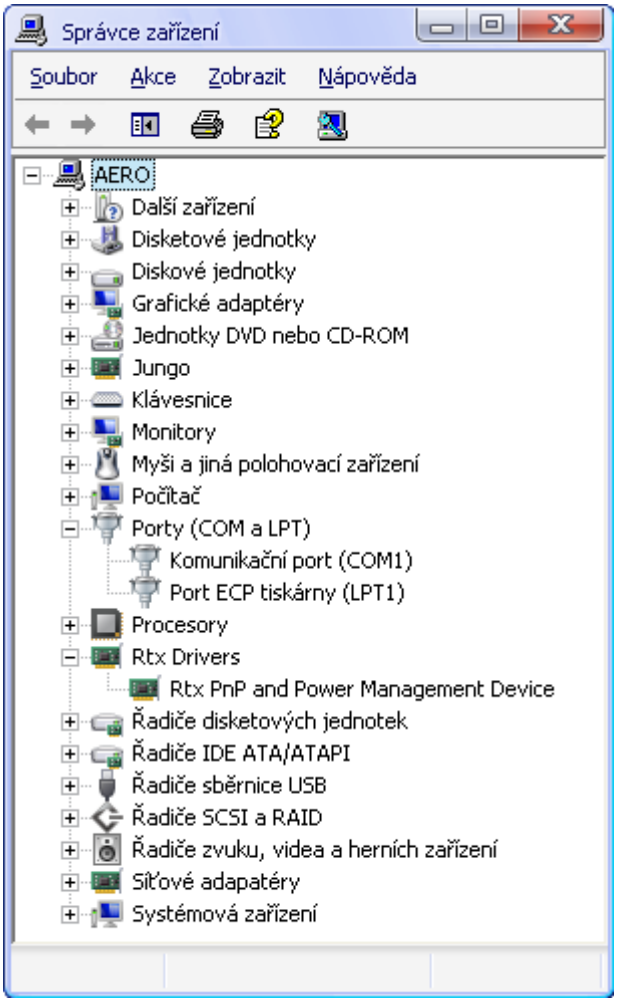
7. Zobrazí se vám dialogové okno k odinstalaci zařízení ze systému. Klikněte na OK k odinstalaci zařízení.
8. V okně Windows Device Manager v menu klikněte na ikonu Scan for hardware changes. Poté se vám zobrazí průvodce přidáním nového zařízení.
9. Zvolte automatickou instalaci a klikněte na tlačítko Next. Průvodce vám nyní nabídne seznam možných ovladačů pro toto zařízení.
10. Zvolte ovladač, který je podepsán OS, nebo ten který byl používán pro zařízení před přesunutím do RTX. A klikněte na tlačítko Next.



Obrázek 46: Zvolení nového ovladače pro zařízení

11. Až dokončí průvodce instalaci zařízení, klikněte na tlačítko Finish.

    Po přesunutí do RTX Properties a kliknutí na tlačítko Refresh se již zařízení zobrazí v seznamu Windows Device.



Obrázek 47: Okno Windows Device Manager a přeinstalované zařízení

Nyní musíte restartovat počítač (v OS Windows XP Professional není restart nutný).

Příloha E. Zdrojové soubory k programům LPT

Main soubor vysílače v prostředí RTX (použití windows funkce pro přerušení).

```
#include       <windows.h>    // knihovna visual studia
#include       <wchar.h>      // knihovna visual studia
#include       <rtapi.h>      // knihovna RTX

#define        CT_PORT_BASE   ((PUCHAR) 0x379)      // nastaveni LPT portu
#define        CT_PORT_RANGE  4                     // nastaveni LPT portu

LARGE_INTEGER  liStart;       // pomocne casove promenne
LARGE_INTEGER  liPeriod;      // pomocne casove promenne
LARGE_INTEGER  liPeriod2;     // pomocne casove promenne
LARGE_INTEGER  time;          // pomocne casove promenne

int            pocitadlo = 0; // pomocne casove promenne
char           znak, prvni;   // pomocne casove promenne

void RTFCNDCL rec(PVOID context)
{
       int            pocitadlo = 0;
       unsigned char  horni, dolni;
       znak = RtReadPortUchar(CT_PORT_BASE);
       prvni = znak;
       do{            // uprava znaku
              dolni = znak;
              dolni = (dolni<<1);
              dolni = (dolni>>4);
              prvni = RtReadPortUchar(CT_PORT_BASE);
              RtSleepFt(&liPeriod2);
              horni = znak;
              horni = (horni>>3);
              horni = horni & 0x0F;
              horni = (horni<<4);
              znak = horni|dolni;
              pocitadlo++;
       }while(pocitadlo == 140);
}

void _cdecl wmain(int argc, wchar_t **argv, wchar_t **envp)
{
       HANDLE         hTimer;        // timer handle
       if(!RtEnablePortIo(CT_PORT_BASE, CT_PORT_RANGE)){
              RtPrintf("RtEnablePortIo error = %d\n",GetLastError());
       }
       RtPrintf("\n\n\nCekam na data...\n\n");

       liStart.QuadPart = 0;
       liPeriod.QuadPart = 5000;     // 50ms
       liPeriod2.QuadPart = 2500;    // 25ms
       if(!(hTimer = RtCreateTimer(NULL,    // security
                                   0,       // stack size - 0 uses default
                                   rec,     // timer handler
                                   NULL,    // NULL context (argument to handler)
                                   RT_PRIORITY_MAX,      // RT max prioritz
                                   CLOCK_FASTEST))){// RTX HAL timer
              RtWprintf(L"RtCreateTimer error = %d\n",GetLastError());
              ExitProcess(1);       // konec pri chybe
       }
       if(!RtSetTimerRelative(hTimer, &liStart, &liPeriod)){      // nastaveni periody
              RtWprintf(L"RtSetTimerRelative error = %d\n",GetLastError());
              RtCancelTimer(hTimer, NULL);
              ExitProcess(1); }     // konech pri chybe
       time.QuadPart = 1000000;
       RtSleepFt(&time);     // doba behu programu
       if(!RtDeleteTimer(hTimer)){   // zruseni hodin
              RtWprintf(L"RtDeleteTimer error = %d\n",GetLastError());
              ExitProcess(1);}
       if(!RtDisablePortIo(CT_PORT_BASE, CT_PORT_RANGE)){  // zavreni portu
              RtPrintf("RtEnablePortIo error = %d\n",GetLastError());}
       ExitProcess(0);       // konec programu
}
```

Main soubor přijímače v prostředí RTX (použití windows funkce pro přerušení).

```
#include       <windows.h>    // knihovna visual studia
#include       <wchar.h>      // knihovna visual studia
#include       <rtapi.h>      // knihovna RTX

#define        CT_PORT_BASE   ((PUCHAR) 0x379)      // nastaveni LPT
#define        CT_PORT_RANGE  4                     // nastaveni LPT

LARGE_INTEGER  liStart;       // pomocne casove promenne
LARGE_INTEGER  liPeriod;      // pomocne casove promenne
LARGE_INTEGER  liPeriod2;     // pomocne casove promenne
LARGE_INTEGER  time;          // pomocne casove promenne

int            pocitadlo = 0;
char           znak, prvni;

void RTFCNDCL rec(PVOID context)
{
       int            pocitadlo = 0;
       unsigned char  horni, dolni;
       znak = RtReadPortUchar(CT_PORT_BASE);
       prvni = znak;
       do{            // priprava znaku a odesilani
              dolni = znak;
              dolni = (dolni<<1);
              dolni = (dolni>>4);
              prvni = RtReadPortUchar(CT_PORT_BASE);
              RtSleepFt(&liPeriod2);
              horni = znak;
              horni = (horni>>3);
              horni = horni & 0x0F;
              horni = (horni<<4);
              znak = horni|dolni;
              pocitadlo++;
       }while(pocitadlo == 140);
}

void _cdecl wmain(int argc, wchar_t **argv, wchar_t **envp)
{
       HANDLE         hTimer;        // timer handle
       if(!RtEnablePortIo(CT_PORT_BASE, CT_PORT_RANGE)){   // otevreni portu
              RtPrintf("RtEnablePortIo error = %d\n",GetLastError()); }
       RtPrintf("\n\n\nCekam na data...\n\n");
       liStart.QuadPart = 0;
       liPeriod.QuadPart = 5000;     // 50ms
       liPeriod2.QuadPart = 2500;    // 25ms
       // zavreni portu
       if(!(hTimer = RtCreateTimer(NULL,    // security
                                   0,       // stack size - 0 uses default
                                   rec,     // timer handler
                                   NULL,    // NULL context (argument to handler)
                                   RT_PRIORITY_MAX,       // nastaveni max priority
                                   CLOCK_FASTEST))){// RTX HAL timer
              RtWprintf(L"RtCreateTimer error = %d\n",GetLastError());
              ExitProcess(1); }
       if(!RtSetTimerRelative(hTimer, &liStart, &liPeriod)){      // nastaveni periody
              RtWprintf(L"RtSetTimerRelative error = %d\n",GetLastError());
              RtCancelTimer(hTimer, NULL);
              ExitProcess(1); }
       time.QuadPart = 1000000;
       RtSleepFt(&time);      // „spaci“ funkce
       if(!RtDeleteTimer(hTimer)){   // zruseni hodin
              RtWprintf(L"RtDeleteTimer error = %d\n",GetLastError());
              ExitProcess(1); }
       if(!RtDisablePortIo(CT_PORT_BASE, CT_PORT_RANGE)){  // zavreni portu
              RtPrintf("RtEnablePortIo error = %d\n",GetLastError()); }
       ExitProcess(0);
}
```

Main soubor přijímače v prostředí windows.

```
#define        _CRT_INSECURE_DEPRECATE(fopen_s)
#include       <stdio.h>              // knihovny visual studia
#include       <stdlib.h>
#include       <dos.h>
```

```
#include       <string.h>
#include       <math.h>
#include       <conio.h>
#include       <time.h>
#include       "DirectIO.h"   // knihovna vytvorena Ph.D. Kacmarem
#define        inportb ReadPortB     // port pro cteni

void main()
{       // nastaveni adresy portu LPT a zakladnich promennych
        int             Port = 0x379;
        int             pocitadlo = 0;
        int             TimeOut = 100;
        char            znak, prvni;
        unsigned char   horni, dolni;
        znak = ReadPortB(Port);
        prvni = znak;
        printf("\n\n\nCekam na soubor\n\n");
        pocitadlo = 0;        // nastaveni pocitadla na 0
        do{
                do{                           // cekani na prvni cast znaku
                        znak = ReadPortB(Port);
                }while(prvni == znak);// uprava pvni prichozi casti znaku
                dolni = znak;
                dolni = (dolni<<1);
                dolni = (dolni>>4);
                prvni = ReadPortB(Port);
                Wait(TimeOut);        // preruseni cteni

                do{                           // cekani na druhou cast znaku
                        znak = ReadPortB(Port);
                }while(prvni == znak);
                horni = znak;         // uprava druhe prichozi casti znaku
                horni = (horni>>3);
                horni = horni & 0x0F;
                horni = (horni<<4);
                znak = horni|dolni; // spojeni dvou casti a vytvoreni znaku
                Wait(TimeOut);        // preruseni na nastavenou dobu
                if(znak == '$'){// overeni jesli prisel ukoncovaci znak
                        return;
                }
                printf("%c", znak);   // vypis znaku do konzoly
                znak = ReadPortB(Port);
                prvni = znak;
                pocitadlo++;
        }while(pocitadlo > 0);
        printf("Kopirovani dokonceno!\n");
        // konec funkce pro cteni znaku z LPT portu
}
```

Main soubor vysílače v prostředí windows.

```
#define _CRT_INSECURE_DEPRECATE(fopen_s)
#include <stdio.h>                    // knihovny visual studia
#include <stdlib.h>
#include <dos.h>
#include <string.h>
#include <math.h>
#include <conio.h>
#include <time.h>
#include "DirectIO.h"                 // knihovna vytvorena Ph.D. Kacmarem
#define outportb WritePortB           // nastaveni LPT portu

void main()
{       // nastaveni pomocnych promennych
        int                     TimeOut = 100;
        int                     Port = 0x378;
        long int                pocitadlo;
        long int                pocetznaku;
        char                    znak;
        char                    soubor[255];
        unsigned char   dolni, dolni_pomocna;
        unsigned char   horni, horni_pomocna;
        FILE                    *f;

        printf("   ***************************************\n");
```

```
        printf("   ********Prenos po paralelnim portu*******\n");
        printf("   ****************************************\n\n");
        printf("Nazev souboru: ");    // nacteni souboru s daty
        scanf("%s", soubor);
        pocitadlo = 0;                // vynulovani pocitadla
        f = fopen(soubor, "r");               // otevreni souboru s daty
        if(!f){                               // overeni spravneho otevreni
                printf("Soubor %s nenalezen\n", soubor);
        }
        fseek(f, 0L, SEEK_END);               // vypocet poctu znaku v souboru
        pocetznaku = ftell(f);
        printf("\nVelikost souboru: %lu KB (%lu B)\n\n", pocetznaku / 1024, pocetznaku);
        printf("Kopiruji soubor, cekejte prosim...\n\n");
        fseek(f, 0L, SEEK_SET);               // navrat na zacatek souboru
        while(feof(f) == 0){  // zacatek funkce posilani dat na LPT port
                znak = getc(f);               // nacteni znaku
                dolni = znak & 0x0F;  // vymaskovani znaku
                if(dolni == horni){
                        dolni_pomocna = ~dolni;
                        dolni_pomocna = dolni_pomocna & 0x10;
                        dolni = dolni|dolni_pomocna;
                }
                horni = znak >> 4;    // priprava casti znaku
                if(dolni == horni){
                        horni_pomocna = ~horni;
                        horni_pomocna = horni_pomocna & 0x10;
                        horni = horni|horni_pomocna;
                }
                WritePortB(Port, dolni);// odeslani dolni casti znaku
                Wait(TimeOut);        // preruseni na nastavenou dobu
                WritePortB(Port, horni);// odeslani horni casti znaku
                Wait(TimeOut);        // preruseni na nastavenou dobu
                printf("%d\n", pocitadlo);
                pocitadlo++;
        }
        znak ='$';                    // priprava na odeslani ukoncovaciho znaku
        dolni = znak & 0x0F;  // uprava ukoncovaciho znaku
        horni = znak >> 4;    // bitovy posun
        WritePortB(Port, dolni);// odeslani dolni casti ukoncovaciho znaku
        Wait(TimeOut);                // preruseni na nastavenou dobu
        WritePortB(Port, horni);// odeslani horni castu ukoncovaciho znaku
        Wait(TimeOut);                // preruseni na urcitou dobu
        printf("Kopirovani dokonceno.\n\n");
// konec funkce pro odesilani znaku na port
}
```

Příloha F. Zdrojové soubory pro program levitace

Výpis funkcí použitých v programu levitace.

```
#include       <windows.h>    // knihovna VS
#include       <stdio.h>      // knihovna VS
#include       <math.h>       // knihovna VS
#include       "rtapi.h"      // knihovna RTX
#include       "karta.h"      // pomocna knihovna

#define        VENDOR_ID_SIZE 2       // pomocne promenne
PCI_COMMON_CONFIG      buffer;        // pomocne promenne
HANDLE hpid = NULL;    // pomocne promenne

double Y = 0;          // pomocne promenne
double W = -1.8;       // pomocne promenne
double U = 0;          // pomocne promenne
double E = 0;          // pomocne promenne
double Ef = 0;         // pomocne promenne
double SumE = 0;       // pomocne promenne
double Efmin = 0;      // pomocne promenne
double  kp = 0;        // pomocne promenne
double  Ti = 0;        // pomocne promenne
double  Td = 0;        // pomocne promenne
double  a = 0;         // pomocne promenne
double  T = 0;         // pomocne promenne
double  DR = 0;        // pomocne promenne

void RTFCNDCL PID(PVOID addr);

////////////////////
// prohledavani PCI slotu dokud bFlag = FALSE
////////////////////

int DeviceSearch(      int vendorID,                // input:  Card Vendor ID
                       int deviceID,                // input:  Card Device ID
                       PCI_SLOT_NUMBER *pSlotNumber, // output: Pointer to slot number
                       PPCI_COMMON_CONFIG  PciData   // output: PCI card information
                       ){
        ULONG          bus;                   // pomocne promenne
        ULONG          deviceNumber;          // pomocne promenne
        ULONG          functionNumber;        // pomocne promenne
        ULONG          bytesWritten;          // pomocne promenne
        BOOLEAN        bFlag = TRUE;  // pomocne promenne
        int            CardIndex = 0;         // pomocne promenne
        pSlotNumber->u.bits.Reserved = 0;     // pomocne promenne
        for(bus = 0; bFlag; bus++){
               for(deviceNumber=0; deviceNumber < PCI_MAX_DEVICES && bFlag;
deviceNumber++){
                       pSlotNumber->u.bits.DeviceNumber = deviceNumber;
                       for(functionNumber = 0; functionNumber < PCI_MAX_FUNCTION;
functionNumber++){
                              pSlotNumber->u.bits.FunctionNumber = functionNumber;
                              bytesWritten = RtGetBusDataByOffset(
                                     PCIConfiguration,// Type of bus data to be
retrieved
                                     bus,                   // Zero-based number of the
bus
                                     pSlotNumber->u.AsULONG,// Logical slot number
                                     PciData,// Pointer to a buffer for configuration
information
                                     0,                     // Byte offset into buffer
                                     PCI_COMMON_HDR_LENGTH        // Length of buffer
                                     );
                              if(bytesWritten == 0){        // Out of PCI buses done
                                     bFlag = FALSE;
                                     break;}
                       if(bytesWritten == VENDOR_ID_SIZE && PciData->VendorID ==
               PCI_INVALID_VENDORID){// No device at this slot number, jump to next slot
                                     break;
                              }
                       if((PciData->VendorID == vendorID) && (PciData->DeviceID ==
               deviceID))
                       return bus;}          // Function Number loop
               }                             // Device Number loop
```

```
        }                                       // Bus loop
        return DEVICE_NOT_FOUND;
}

////////////////////
// MAIN funkce
////////////////////

int main(int argc, char *argv[])
{
        PPCI_COMMON_CONFIG    pciData = (PPCI_COMMON_CONFIG)       &buffer;
        PCI_SLOT_NUMBER       slotNumber;           // pomocne promenne
        ULONG                 busNumber;            // pomocne promenne

        LARGE_INTEGER         length;               // pomocne promenne
        LARGE_INTEGER         start;                // pomocne promenne
        LARGE_INTEGER         wait;                 // pomocne promenne
        HANDLE                timerh;               // pomocne promenne

        if(argc != 7){                // overeni poctu vstupnich argumentu
                RtPrintf("not enought arguments...\n");
                ExitProcess(0); }

        kp = atof(argv[1]);           // prevod argumentu na cisla
        Ti = atof(argv[2]);           // prevod argumentu na cisla
        Td = atof(argv[3]);           // prevod argumentu na cisla
        a = atof(argv[4]);            // prevod argumentu na cisla
        T = atof(argv[5]);            // prevod argumentu na cisla
        DR = atof(argv[6]);           // prevod argumentu na cisla

        printf("kp = %f\nTi = %f\nTd = %f\na = %f\nT = %f\nW = %f\nDR = %f\n", kp, Ti,
Td, a, T, W, DR);     // vypis argumentu do souboru

        // funkce hledajici PCI kartu
        busNumber = DeviceSearch(VENDOR_ID, DEVICE_ID, &slotNumber, pciData);

        // vysledek hledani...
        if(busNumber == DEVICE_NOT_FOUND){
                RtPrintf("Measure card wasn't found on this PC.");
                if(hpid)
                        RtCloseHandle(hpid);
                ExitProcess(0);}
        // kdyz nebyla karta nalezena
        if(!(busNumber == DEVICE_NOT_FOUND)){
                RtPrintf("Measure card sucesfuly founded and preparing process in
progress.");
                inicialization();// inicializace karty
        }
        // vytvoreni hodin
        if(!(timerh = RtCreateTimer(NULL,
                                    0,
                                    PID,
                                    NULL,
                                    RT_PRIORITY_MAX,
                                    CLOCK_FASTEST))){
                RtPrintf("Fatal Error: Could not get timer handle.  GetLastError = %d\n",
GetLastError());
                if(hpid)
                        RtCloseHandle(hpid);
                return(ERROR_OCCURED);
        }
        // nastavovani periodicke funkce
        length.QuadPart = (LONGLONG)(T*10000000);
        start.QuadPart = (LONGLONG)(T*100000000);
        if(!RtSetTimerRelative(timerh, &start, &length)){
                RtPrintf("Fatal Error: Could not initialize timer.  GetLastError = %d\n",
GetLastError());
                RtCancelTimer(timerh, NULL); // ruseni hodin
                if(hpid)
                        RtCloseHandle(hpid);
                return (ERROR_OCCURED);
        }
        // „spaci“ funkce, spi během regulace
        wait.QuadPart = (LONGLONG)(DR*10000000);
        RtSleepFt(&wait);

        if(!RtCancelTimer(timerh, NULL)){ // zruseni hodin
```

```
                RtPrintf("Error: Could not cancel timer.  GetlastError = %d\n",
GetLastError());
                if(hpid)
                        RtCloseHandle(hpid);
                return ERROR_OCCURED;
        }
        // na konci regulace nastavi na vystup -5V
        RtWritePortUlong(DACReg0l, 0x0);
        RtWritePortUlong(DACReg0h, 0x0);
        RtPrintf("\nRegulation finish succesfully.\n");
        RtDeleteTimer(timerh);
        RtCloseHandle(hpid);
        return (NO_ERRORS);
}

////////////////////
// funkce regulace
////////////////////

void RTFCNDCL PID(PVOID addr)
{
        PPCI_COMMON_CONFIG  pciData = (PPCI_COMMON_CONFIG) &buffer;

        ULONG   low = 0;        // pomocna promenna pro cteni z bufferu
        ULONG   high = 0;       // pomocna promenna pro cteni z bufferu
        LONG    out = 0;        // pomocna promenna pro zapis
        LONG    maxW = 0xFFF;
        LONG    Ul = 0;         // pomocna promenna pro zapis

        double  P = kp;         // vypocet P
        double  I = kp * (T/Ti);// vypocet I
        double  D = kp * (Td/T); // vypocet D
        // nastaveni merici sekvence
        RtWritePortUlong(CWReg, SCWReg);
        while((RtReadPortUlong(StatusReg)&0xFF) != 0x0);
        RtWritePortUlong(SWTrigReg, SSWTrigReg);
        while((RtReadPortUlong(StatusReg)&0xFF) != 0x0);
        low = (RtReadPortUlong(REG_LO) & 0xFF);      // nacteni dolni casti
        high = (RtReadPortUlong(REG_HI) & 0xFF);     // nacteni horni casti
        Y = ((high<<8)|low);                  // spojeni nactenych dat
        RtWritePortUlong(CWReg, XCWReg);      // STOP merici sekvenci

        Y = ((10*Y/0xFFFC)-5);        // prevod 14bitove hodnoty na napeti ve voltech
        printf("\nY\t%f", Y);         // vypis hodnoty do souboru
// vlastni vypocet regulace
        E = W - Y;
        Ef = (1-a)*Efmin + a*E;
        SumE += E;
        U = P*E + I*SumE + D*(Ef - Efmin);
        printf("\nU\t%f", U); // vypis hodnoty do souboru
        Efmin = Ef;
        U = (U*0xFFF/10)+(0xFFF/2);   // vzorec pro prepocet z napeti na 12 bitovou
hodnotu
        U = (int)(U + 0.5); // zaokrouhleni
        Ul = (LONG)U;
        if(Ul > maxW){ // zapis na vystup, kdyz je vystup vetsi nez mozny rozsah
                RtWritePortUlong(DACReg0l, 0xff);
                RtWritePortUlong(DACReg0h, 0xf);
        }else{
                out = (Ul & 0xff);
                RtWritePortUlong(DACReg0l, out); // zapis na vystup
                RtWritePortUlong(DACReg0h, Ul>>8);
        }
}
```

Výpis funkce pro nastavení měřící karty PCA 7428AS, v klientovi.

```
void inicialization(){ // funkce nastavujici merici kartu pred spustenim
 PPCI_COMMON_CONFIG  pciData = (PPCI_COMMON_CONFIG) &buffer;
 RtWritePortUlong(DACReg0l, 0xff); // nastaveni DA prevodniku dolni cast
 RtWritePortUlong(DACReg0h, 0xf); // nastaveni DA prevodniku horni cast
 RtWritePortUlong(CWReg, XCWReg); // pripadne ukoncene probihajiciho mereni
 RtWritePortUlong(BufferPageReg, 0x0); // vynulovani bufferu
 RtWritePortUlong(ScanADCReg, SScanADCReg); // nastaveni analogoveho vstupu 0x20 – vstup
0 zesileni +-5V
```

```
RtWritePortUlong(ScanChanReg, SScanChanReg); // nastaveni poctu skenovacich kanalu 0x1 – skenovan jeden analogovy vstup
 RtWritePortUlong(ADCDelayEnReg, SADCDelayEnReg); // pro modifikaci implicitnich casovych pomeru merici sekvence 0x0 – nemodifikuj
 RtWritePortUlong(ADCDelayReg, 0x0); // nastaveni zpozdeni pro kanaly se spozdenim 0x0 - nemodifikuje
 RtWritePortUlong(ScanCNTReg, SScanCNTReg); // aktivace zaznamu citacu do datoveho zasobniku 0x0 - neaktivuj
 RtWritePortUlong(SetCNT0Regl, SSetCNT0Reg); // definuje pocatecni hodnotu citace 0x0 - 0
 RtWritePortUlong(SetCNT0Regh, SSetCNT0Reg); // definuje pocatecni hodnotu citace 0x0 - 0
 RtWritePortUlong(SetCNT1Regl, SSetCNT1Reg); // definuje pocatecni hodnotu citace 0x0 - 0
 RtWritePortUlong(SetCNT1Regh, SSetCNT1Reg); // definuje pocatecni hodnotu citace 0x0 - 0
 RtWritePortUlong(CfgCNTReg, SCfgCNTReg); // konfigurace vstupnich obvodu citacu 0x0 – citac blokovan
}
```

Příloha G. Spouštěcí rutina měřící karty

Při tvorbě spouštěcí rutiny pro program řízení levitace se musely respektovat postupy obsluhy měřící karty.

Pro spuštění byl zvolen způsob programového spuštění měření. Postup nastavení registrů byl následující:

1	nastavení scanovací logiky	ScanADCReg, ScanChanReg, ScanCNTReg, SetCNT0Reg, SetCNT1Reg, CfgCNTReg, ADCDelayEnReg, ADCDelayReg
2	zapsat data zvoleného pracovního režimu	CWReg
3	počkat a vyhodnotit provedení inicializace desky	StatusReg == 0
4	spustit měřicí sekvenci zápisem	SWTrigReg
5	vyhodnotit konec měřicí sekvence	StatusReg == 0
6	načíst data z datového zásobníku	
7	cyklické opakování měření	podle bodů 4 až 6
8	ukončit měření zápisem	CWReg

Příloha H. Technické parametry karet PCA – 7228AS a PCA – 7428AS

A/D PŘEVODNÍK	
Počet a typ analogových vstupů	8x S.E. (možnost rozšíření na 32 S.E. pomocí OPT-832 nebo OPT-830B)
Rozlišení analogových vstupů	12 bitů (AD7895) – PCA-7228AS 14 bitů (AD7894) – PCA-7428AS
Vstupní rozsahy	±10V, ±5V, ±2.5V, ±1.25V, ±0.635V, ±0.3125V
Chyba rozsahu	±0.1% typ. (lze kalibrovat)
Nesymetrie	±0.1% typ. (±0.2% max.)
Programovatelné zesílení	1x, 2x, 4x, 8x, 16x, 32x
Chyba zesílení	±0.05% typ. (±0.15% max)
Vstupní impedance	10MOhm typ.
Maximální vstupní napětí	±24V (trvale) ±50V (10ms max.)
Logika spouštění	
Zdroje spouštění A/D převodníku	interní časovač, softwarový start, externí signál TTL (sestupná hrana)
Rozsah při spouštění časovačem	30.5 Hz ~ 100 kHz
Rozsah při spouštění externím signálem	0 Hz ~ 100 kHz
Rozsah při softwarovém spouštění	závisí na operačním systému celková doba převodu je cca 100µs + doba vlastního měření uvedená dále
Doba A/D konverze	10 µs max. (zesílení 1x ~ 8x) 13 µs max. (zesílení 16x) 18 µs max. (zesílení 32x) +2 µs max. (navýšení OPT-832) +40 µs (v režimu průměrování)
Doba zpracování dat čítače	10 µs max. / 1 čítač (měřeny pouze čítače) 6 µs max. / 1 čítač (měřen alespoň 1 AIN) umožňuje konfiguraci prodlev nezávisle pro každý rozsah
D/A převodníky	
Počet a rozlišení D/A převodníků	2x 12 bitů
Výstupní rozsahy	0~5 V, ±5 V (volba spínačem)
Chyba rozsahu	±0,1% typ. (±0,2% max.)
Nesymetrie	±0,1% typ. (±0,2% max.)
Doba ustálení analogového výstupu (±0,5% FSR)	5 µs typ. (15 µs max.)
Výstupní impedance	< 1 Ohm typ.
Zatěžovací impedance	500 Ohm min. Výstupy D/A převodníků jsou odolné proti trvalému zkratu proti GND. Přivedením napětí mimo rozsah ±12 V dojde k nevratnému poškození obvodů.

Čítač, časovač	
Počet a rozlišení čítačů	2x 16 bitů
Pracovní frekvence	2 MHz max. (střída signálu 1:1)
Pracovní úrovně	TTL/HCMOS (sestupná hrana)
Další funkce čítačů	softwarové hradlování, externí hradlování s volitelnou úrovní
Přenos dat do PC	synchronní s daty analogových vstupů
	Vstupní obvody čítačů jsou odolné proti přepětí ±24 V; přivedením napětí mimo povolený rozsah dojde k nevratnému poškození obvodů.
Obvody přerušení	
Zdroj přerušení	zásobník 256 B, konec měřicí sekvence zásobník 256 B, zaplnění 128 B zásobník 64 kB, zaplnění 256 B zásobník 64 kB, zaplnění 512 B zásobník 64 kB, zaplnění 2 kB zásobník 64 kB, zaplnění 8 kB zásobník 64 kB, zaplnění 32 kB Režimy "zásobník 256 B" jsou u PCA-7228A implementovány z důvodu zpětné kompatibility s řadou PCA-7208A/7408A.
Digitální porty	
Počet vstupů	8 (TTL komp.)
Počet výstupů	8 (TTL komp.)
Zatěžovací impedance výstupů	500 Ohm min.
Ostatní údaje	
I/O a MEM adresa	přiřazena PnP PCI BIOSem
IRQ kanál	přiřazen PnP PCI BIOSem
Napájení a proudový odběr	
+5V	350 mA max.
+12 V, -12 V	60 mA max.
Rozměry desky	cca 90 x 125 mm
Použité konektory	Cannon 25 - vidlice Cannon 9 - vidlice
Pracovní teplota	0° ~ 65° C
Skladovací teplota	-20° ~ 80° C
Relativní vlhkost	10% ~ 90%, bez kondenzace
Doporučená délka vodičů	2 m max.
Datový zásobník	64kB (dvou-brannou RAM)
Sběrnice	standardní PCI (32 bitů, 33 MHz, 5V)